Escuela Superior de Ingeniería

Programa de Doctorado en Ingeniería y Arquitectura

Tesis Doctoral

Estudio e implementación de algoritmos de fusión sensorial para sensores pulsantes y clásicos con protocolo AER de comunicación y aplicación en sistemas robóticos neuroinspirados

---

Memoria presentada para aspirar al grado de doctor
(Mención internacional)

Autor: **Fernando Pérez Peña**
Los profesores ponentes y directores:

Dr. Arturo Morgado Estévez
Profesor Titular del Dpto. Ing. en Automática, Electrónica, Arquitectura y Redes de Computadores
Universidad de Cadiz

Dr. Alejandro Linares Barranco
Profesor Titular del Dpto. de Arquitectura y Tecnología de Computadores
Universidad de Sevilla

Cádiz, Octubre 2014

University of Cadiz

A thesis submitted for the degree of Doctor of Philosophy (PhD) in Engineering and Architecture

Study and implementation of sensory fusion algorithms: spiking and classic sensors using AER communication protocol and application to neuro-inspired robotic platforms

Author:
Fernando Pérez Peña

Advisors:
Dr. Alejandro Linares Barranco
Universidad de Sevilla
Dr. Arturo Morgado Estévez
Universidad de Cadiz

Cadiz, October 2014

El Dr. D. Alejandro Linares Barranco como Director de la Tesis Doctoral titulada "Estudio e implementación de algoritmos de fusión sensorial para sensores pulsantes y clásicos con protocolo AER de comunicación y aplicación en sistemas robóticos neuroinspirados" realizada por D. Fernando Pérez Peña en el Departamento de Ing. en Automática, Electrónica, Arquitectura y Redes de Computadores de la Universidad de Cádiz, informa que la tesis reune todos los requisitos necesarios para el depósito y posterior defensa.

Fdo. Alejandro Linares Barranco

Zurich, 4 de Septiembre 2014

El Dr. D. Arturo Morgado Estévez como Director de la Tesis Doctoral titulada "Estudio e implementación de algoritmos de fusión sensorial para sensores pulsantes y clásicos con protocolo AER de comunicación y aplicación en sistemas robóticos neuroinspirados" realizada por D. Fernando Pérez Peña en el Departamento de Ing. en Automática, Electrónica, Arquitectura y Redes de Computadores de la Universidad de Cádiz, informa que la tesis reune todos los requisitos necesarios para el depósito y posterior defensa.

Fdo. Arturo Morgado Estévez

Cádiz, 4 de Septiembre 2014

Wherever you are, thanks Mum

Acknowledgements

This piece of paper is small to provide all the thanks I have to give, but I will try to do my best.
I would like to thank both of my advisors: Dr. Arturo Morgado Estevez and Dr. Alejandro Linares Barranco. You have given me, from the beginning, all the guidance and support that one can expect. Thank you for standing by me every moment, even weekends and holidays.
I must thank Maria for her pacients, love and support. I am deeply grateful to her because she always understands me even during the frustrating and crazy moments during these last three years. You are part of this thesis.
I would also like to thank all my family: Dad, brothers, sister, Juanma, Rocio, Bea, Maria Jesus and the new members: Lucia and Alejandro. You have given me the best support and the most valuable advice when I needed it. Many times, you have shown me how to continue and helped me to calm down.
I would also like to thank to "Villa's family". I will never forget the long chat with Maria at the begining of this phase of my life, the trip to London with Lola and the great moments during the summer. Of course, thank you to Juan Villa, who always gave me the right comment. We miss you.
My thanks also goes to my friends Alvaro, Ana, Jorge, Alicia, Pavón, Abelleira, Jose Maria and Pablo. You guys have given me great moments of laughter and disconnect from the thesis. Special thanks to Pablo for the cover design of this thesis.
Also, thank you to Dr. Jose Antonio Alvarez Gomez, MD, Dr. Maria Jesus Sanchez del Pino, MD and Ana Collantes Gonzalez, MD. They reviewed the bilogical part of chapter three and gave me valuable feedback from a medical prospective.
Likewise, I must thank my ATC colleagues from the University of Cadiz

(special thanks to Manuel, Carlos, Mercedes and Nestor), ATC colleagues from the University of Seville (special thanks to Angel, Manu, Paco, Gabriel, Rafa and Elena) for their support, advice and all the time we have spent together. Thank you to Jose Luis Muñoz Lozano and Juan Lopez Coronado, for hosting me at the Politechnics University of Cartagena.
Also, thank you to Dr. Elisabetta Chicca and her NBS group for hosting me at the University of Bielefeld in Germany. Thank you to Glicka and Martin ("my German family"), Marius, Tim, Harsha, Stefan and Frank for helping me there with everything. It was really easy to adapt to living in a foreign country with you guys.
I would like also to thank Mark Williams, my friend and English teacher, during the last year. Thank you very much Mark for proof reading the thesis.
I would like to dedicate this thesis to my Mum.
Finally, I would like to thank the following programs for providing funding that allowed me to make this thesis: the University of Cadiz fellowships program, Research Plan of the University of Cadiz, VULCANO and BIOSENSE projects (Spanish national research projects) and the DAAD (German Academic Exchange Service).

Resumen

La presente tesis doctoral tiene como objetivo analizar, diseñar, simular e implementar un modelo que siga los principios de funcionamiento del sistema nervioso humano para imitar los movimientos dirigidos a un objetivo. El campo de investigación en el que se enmarca la tesis es la ingeniería neuromórfica. Dicho campo nace a finales de los años ochenta y persigue el desarrollo de dispositivos electrónicos que, basados en la neurona como unidad básica, puedan ser empleados para desarrollar alguna actividad específica: toma de decisiones, procesado de imágenes, aprendizaje, etc.
Se propone un modelo de controlador neuro-inspirado basado en los algoritmos clásicos VITE y FLETE a implementar en dos clases de hardware: una FPGA y un entorno multi-chip VLSI. Para ello, teniendo en cuenta cómo se realiza un movimiento hacia objetivo en los seres humanos, se transforman estos algoritmos considerando las restricciones propias de cada uno de los entornos hardware a emplear: bloques de procesamiento por eventos descritos en VHDL para la FPGA y neuronas LIF en los chips VLSI. Para una exitosa traslación del algoritmo VITE bajo las restricciones de la FPGA, se diseña, simula e implementa un nuevo bloque de procesamiento por eventos: Bloque GO.
En contra, para una correcta y biológica traslación del algoritmo VITE bajo los requisitos VLSI, se estudia, modela, simula e implementa la co-activación de los canales NMDA que guardan relación con la actividad previa detectada en los ganglios basales.
Una vez que el modelo está definido para ambas plataformas, se procede a la simulación de este usando el entorno Simulink de Matlab para la FPGA y el simulador Brian para los chips VLSI.
Finalmente, se presentan los resultados de implementación del algoritmo SVITE en lazo abierto (en ambas plataformas) y cerrado (FPGA) aplicados

a una plataforma robótica y conectados usando el bus AER con unos excelentes resultados en cuanto a consumo de potencia y recuros y paralelismo con lo estudiado para los movimientos biológicos. Los resultados muestran un correcto funcionamiento cuando se trata de alcanzar o seguir un objetivo suministrado por una retina AER. De esta forma, se completa una cadena de procesado por eventos desde el sensor hasta los efectores finales de una forma neuro-inspirada.
También, se presenta una variante del algoritmo, para la plataforma FPGA, en el que se incluyen elementos aleatorios que modelan la variabilidad en la respuesta neuronal. Los resultados obtenidos para esta variante, muestran un resultado similar al comportamiento totalmente determinista. Se demuestra la posibilidad de incluir el controlador en un entorno ruidos y/o aleatorio.
Por último, se confirma el uso de PFM como la modulación más indicada en entornos biológicos ya que permite suministrar los eventos directamente a los motores. Además, se logra que el sistema no se vea afectado por eventos espúreos o no deseados.
Los novedosos resultados alcanzados en la plataforma VLSI, este ha sido el primer intento de controlar una plataforma robótica usando diseño *subthresold* en chips de bajo consumo, pretenden sentar las bases del diseño de controladores neuro-inspirados.

Contents

List of Figures

List of Tables

Chapter 1

Introduction: Motivations and objectives of the thesis

This chapter sets out the motivation and the objectives of this thesis. A section where the motivations are expressed comes first. Then, the objectives are listed. Finally, the structure of the thesis document is described.

1.1 Motivations

The framework of this research is the Neuromorphic engineering field. The objective of this field to develop hardware devices based on the principles of the nervous system. Eventually, these devices are used to build large architectures. Nowadays, the field has already developed sensors and general purposed chips where a big amount of neurons and synapses are allocated. Nevertheless, there is a gap to bridge between all the event-based sensory and processing with the actuation part across the chips designed. Thus, our research aims at developing a full motor controller closely inspired by the control mechanisms found in the human nervous system. This research has high potential for improving understanding of biological motor control, developing novel controlling techniques and eventually substantially reducing power consumption. This is one of the motivations of the thesis: to try to improve the classical robotics control techniques by applying bio-inspired methods.

On the other hand, Why can't we think about a hypothetical connection with the human nervous system by using these devices? This is the objective of the neuroprosthetic field which is one of the motivations for this thesis: to try to enhance the knowledge we have about the brain mechanism for motor controlling and its interface with a real robotic platform. This is one of the main motivations of the thesis.

1.2 Objectives

The main objective for this doctoral dissertation is to develop a full neuro-inspired motor controller using neuromorphic hardware that is as close as possible to an intended biological movement. Thus, to achieve a successful controller, the objectives to cover are:

- Study and analyse the bio-inspired algorithms suitable to be implemented using neuromorphic hardware.
- Propose a neuroinspired model: trajectory and forces controller. Translate the algorithms found into the spikes paradigm using the constraints of the neuromorphic hardware to use: FPGA and VLSI chips.
- Analyse, design and test the blocks or circuits needed for the spike-based model proposed using simulators (different per each type of neuromorphic hardware). This objective is very important because the success here led to a success in the hardware implementation.
- Simulate the whole model to achieve the desired behaviour.
- Implement the designed model using neuromorphic hardware and test it using a real robotic platform. Code the VHDL and Python scripts necessary to run the tests. The neuromorphic hardware to use is FPGA and VLSI chips.
- Include in the model, random components to be as closer as possible to a biologically behaviour. Implement this algorithm and make a comparison with the non-random one designed.
- Include a vision sensor to achieve a full spike-based controller: from the sensor to the actuation.
- Study and propose a method to select the parameters to drive the motors of the robotic platform.

1.3 Thesis Structure

The dissertation has a total of six chapters and two appendixes. It follows as is listed:

- Chapter two: Neuromorphic engineer
  This chapter will focus on the field of the thesis: the neuromorphic engineer. First, the objective of this community will be stated; then, the neuromorphic hardware and the related works are listed and explained. There follows a comparative between two hardware platforms; eventually, the Address Event Representation protocol is presented.

- Chapter three: Intended biological movements
  This chapter will focus on how the movements are performed by humans under a biological approach and what models are available in previous works, to implement with electronic devices, the human control system for motion generation.

- Chapter four: Event-based processing model
  This chapter will focus on how the algorithms previously explained can be adapted for use with spiking neuron models. It includes a first part where the algorithms are prepared to be used within the spike paradigm. The event-processing blocks needed to implement the model are presented and explained. Then, a second part will describe the final model proposed to implement them, using the following platforms: FPGA and VLSI chips.

- Chapter five: Results and Discussion
  This section will present the results achieved by the research within this thesis. Two main sections are presented: simulated results and hardware results. Then, each of this section includes several subsections where the graphs are shown and explained. Finally, a discussion section is presented.

- Chapter six: Summary, conclusions and future work

- Appendix A: PWM Vs. PFM
  This appendix comprises of a report about Pulse Width Modulation (PWM) and Pulse Frequency Modulation (PFM) modulation used to drive motors using spiking neural networks.

- Appendix B: Publications
  This appendix lists the publications that this Ph.D has produced.

Chapter 2

Neuromorphic engineer

This chapter will focus on the field of the thesis: the neuromorphic engineer. First, the objective of this community will be stated; then, the neuromorphic hardware and the related works are listed and explained. There follows a comparative between two hardware platforms; eventually, the Address Event Representation protocol is presented.

2.1 Introduction

The main goal of the neuromorphic engineering research field is to develop hardware devices based on the principles of the human nervous system [9–12]. The term "Neuromorphic Engineering" (NE) was first coined by Caver Mead in the late eighties [13]. He started mimicking the behaviour of neuron cells by using Very-Large-Scale-Integration (VLSI) chips based on CMOS transistors.

From the very beginning it has been known that the nervous system uses spikes or action potentials to carry the information across the organism [4]. The outstanding behavior of those systems suggests mimicking them into electronic devices based on interconnected spiking neural networks: the third generation of neural networks [14]. Therefore, the challenge of the neuroengineering community is to create architectures of neuromorphic chips with the same properties of human neural system: low power consumption, compact size and scalability. The main objective is to be as closer as possible to

these principles to enhance the functionality of systems in fields like robotics, neuroengineering, etc.

Nowadays, the NE community is a large one working in many countries all around the world. There are two annual workshops where they meet and share ideas and new projects start: Telluride Cognitive Neuromorphic Workshop [15] and CapoCaccia Cognitive Neuromorphic Workshop [16]. The usual procedure is the close study of any biological nervous system concept which is suggested to be implemented in silicon. Then an analog model is designed and simulated until the expected behaviour is reached and eventually manufactured. The final product will be called a neuromorphic device. The information flowing in these devices consists of spikes or graduated potentials like those used by neurons to carry information across the organism. The main features of these full-custom chips are noise due to thermal fluctuations, high speed/bandwidth usage and computation with continuous values as information.

There are many European projects focused on building computing systems which exploit the capabilities of these devices: Brain-inspired multiscale computation in neuromorphic hybrid systems (BrainScale), SpiNNaker and the Human Brain Project (HBP) as examples. One of the main challenges is to select which devices to use and how to integrate them to produce functional elements.

In the nineties, Sivilotti [17] defined the communication protocol to be used between those devices: Address Event Representation (AER). AER enables the communication of thousands of neurons from one chip to another. Within AER, each neuron is given an address to be identified within the architecture. All the devices were expected to be connected with that AER bus [18].

2.2 Neuromorphic Hardware

The most popular options to implement the neuromorphic devices into hardware are these three: the use of a full custom VLSI design application-specific integrated circuit (ASIC) [19], a Field Programmable Gate Array (FPGA) [20, 21] or a Field-Programmable Analog Array (FPAA) [22].

The neuromorphic hardware available comprises of fields such as sensors: vision sensors [23–27], cochlea sensors [28, 29], olfaction sensors [30] and chips of general purpose [31, 32]. The latest analog systems are Neurogrid [33], Sensemaker or the human brain project [34]; the digital ones are Spinnaker [35], Neurovision (with military funding), the wafer-scale neuromorphic hardware developed under FACETS project [36] and the chip developed by IBM, the Truenorth [37].

However, a challenge today is to bridge the gap between sensors and large architectures, including an end-effector like a robotic platform, to reach an accurate spiking motor control. Nowadays, with the described real-time neuromorphic systems available, the research is well suited for exploring biologically inspired motor control algorithms. There is not a high number of publications related to bio-inspired motor controlling using neuromorphic hardware; there are some works where the Dynamic Vision Sensor [24] is used to create a pencil balancing robot [38], to close the control loop using Spinnaker [39] or to guide a robotic arm [40]. Recently, a classical industrial approach such as Proportional-Integral- Derivative (PID) control has been proposed to develop a neuromorphic motor controller [41, 42]. Nevertheless, this approach is developed under industrial constrains that might not fit with neuromorphic engineering goals. Recent work presented in [43] and [44] describes the use of neuromorphic hardware to control the robot motor torques and to control a small robotic arm, respectively.

In addition to these hardware works, [45, 46] present software works where cognition and learning tasks are developed by spaun: a 2.5-million-neuron brain model [47, 48].

Finally, in the field of neuroprosthetics along with the NE, some works have been done: in [49], an interface between neuromorphic hardware and the spinal cord is made, the HBP aims to build a virtual laboratory for neuroprosthetic prototyping [34], in [50] the first example of a neuromorphic device that can replace some functions of a component of the mammalian central nervous system in vivo is presented, [51] presents a visual neuroprosthetic where AER is used and [52] gives a review of the state of research on visual neuroprosthesis connected to the visual cortex.

In our objective to generate intended movements towards a target in a biological way with electronic devices, we have to deal with several problems when implementing the spike-based algorithm:

- How to consider the information: in these systems each neuron fires a spike when it reaches a specific threshold in a completely asynchronous way. There are several ways to encode these spikes; for example, the rate coded [53]: when the excitation is low, the spike rate is low and thus the time between spikes is high; however, when the signal excitation increases, the inter-spikes interval time (ISI) decreases, while the spike rate increases. Consequently, the information is codified as the firing rate or frequency.

- The way to implement this architecture: we have implemented it into a FPGA and a VLSI chips setup. The FPGA system is, apparently, not an asynchronous system but the clock frequency of these digital systems is high enough to allow us to consider an asynchronous behavior for the neurons.

- The other problem is related to the manner of holding communication between different neuromorphic devices. As has been stated, the AER protocol will be used.

2.3 Comparative between the VLSI and FPGA hardware in use

The main features for these full-custom chips are noise due to thermal fluctuations, high speed/bandwidth usage and computation with continuous values as the information. Conversly, architectures based on FPGAs provides stability, fast development times, low noise and high precision properties. Although a clock is inherent to digital designs, the information is in the firing rate, which can be an accurate analog approach.

The VLSI chips are based on the Leaky Integrate and Fire (LIF) neuron model. This neuron model is made of CMOS transistors and the chips include a high number of these neurons and synapses to connect them. The dynamics of the membrane potential in the LIF neuron are described by a single first-order linear differential equation: Equation (2.1). When the

neuron is being excited, the membrane potential of the neuron increases until it reaches its threshold, then, a spike will be fired. Immediately after the firing, the membrane potential is reset to the resting potential value. If the neuron is receiving the effect of an inhibitory synapse, the membrane potential decreases [1]. Figure 2.1 shows this behaviour.

$$\tau_m \times \frac{d}{dt}\upsilon(t) = -\upsilon(t) + R \times I(t) \tag{2.1}$$

Where $\upsilon(t)$ represents the membrane potential at time t, τ_m is the membrane time constant and R is the membrane resistance. This equation describes a simple resistor-capacitor (RC) circuit where the leakage term is due to the resistor and the integration of I(t) is due to the capacitor, that is parallel to the resistor. The spiking events are not explicitly modelled in the LIF model [54].

The chips are general purpose since many parameters can be tuned. Usually, there are other boards in charge of generating the currents to inject the chips in order to achieve a desired value for each parameter; these currents are called biases. To set all these biases, these chips are configured using a software framework code in Python [55].

The FPGA is based on the Integrate and Generate (I&G) neuron model [41] I&G neuron includes one pre and postsynaptic connection and an integrator which computes the ongoing spikes. It models the activity level of the neuron; its firing rate depends on the integrated value. In its basic model, the integrated value is updated for each incoming spike by an increase or a decrease of the membrane potential, modelled by a counter, depending of the polarity of the received spike. If the spike is positive the membrane potential is increased, but it is decreased when the spike is negative. Current membrane potential is continuously translated into an output stream of spikes. This model is similar to the LIF model, but it allows modifying of the distribution over time of the outgoing spikes. The behaviour is currently described using VHDL.

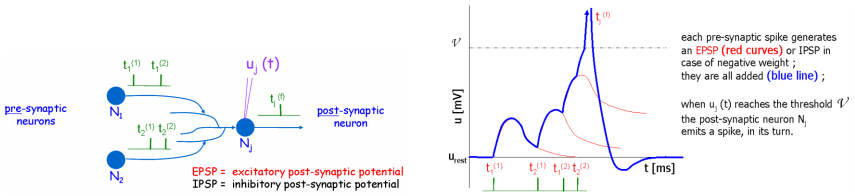

Figure 2.1: A model of spiking neuron: N_j fires a spike whenever the weighted sum of incoming EPSPs generated by its pre-synaptic neurons reaches a given threshold. The graphic (right) shows how the membrane potential of N_j varies through time, under the action of the four incoming spikes (left). Taken from [1].

2.4 Address Event Representation (AER)

One of the problems faced when one tries to integrate and implement large and complex neuromorphic architectures, based on neuromorphic devices, is the communication between them: it is not easy to distinguish which neuron of what device is firing a spike. To solve this problem, Address Event Representation (AER) protocol is used [17].

AER was proposed to solve this communication between neuromorphic devices. It tries to mimic the structure and information coding of the brain. Like the brain, AER will let us process information in real time, by implementing simple spike-based operation at the time each spike is produced or received.

AER maps each neuron with a fixed address which is transmitted through the interconnected neuron system (Figure 2.2). Usually, a neuromorphic chip is designed so that it is continuously sending information about all of its neuron's excitation levels to the system. Thus, connecting several devices with an AER bus, all neurons of a layer are continuously sharing their excitation with the other layers through bus connections; this information can be processed in real time by a higher layer. Just by adding chips to the bus, it is possible to enlarge the system. That is one of the most important reasons for using AER, i.e. the scalability allowed by new connections. Since each chip has an internal arbiter to access the AER bus [56], real time is

limited by the digital clock.

Using the AER protocol, the information can be processed in real time which is an important factot in motor control. This is the other reason for using it: the intrinsic speed behind the spike-based philosophy. The initial AER bus used was an asynchronous parallel one [57] and new studies propose a serial protocol connection [58–60].

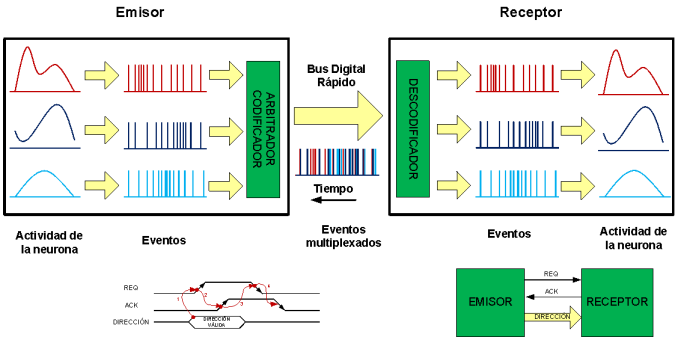

Figure 2.2: Address Event Representation protocol. Taken from [2]

2.5 NE field future

During a very recent event in Melbourne (Australian Neuromorphic Engineering Workshop, June, 2014) where the main researchers of the NE field met, Ralph Etienne-Cummings suggested some present and future possibilities for the neuromorphic engineering field: Can one build something that looks like the brain with all of this neuromorphic hardware? With this hardware, is it possible to build a system that safe power, use less resources or improve some robotic applications? What does NE tell us about how the brain works? Also, Giacomo Indiveri points out that the process of building neuromorphic architecture could enable us to understand more features about the brain. Finally, as a present question: What are the biological systems available which engineers can implement or can create with the current technology?

Chapter 3

Intended biological movements

Biological section of this chapter reviewed by:
Ana Collantes González, MD (Radiologist, Puerta del Mar Hospital, Cádiz, Spain)
Jose Antonio Álvarez-Gómez, MD, Ph.D (Head Anesthesia Department, University Santa Lucía Hospital, Cartagena, Spain)
María Jesús Sánchez del Pino, MD, Ph.D (Biomedicine, Biotechnology and Public Health Dept., University of Cádiz, Spain)

This chapter will focus on how the movements are performed by humans under a biological approach and what models are available in previous works, to implement with electronic devices, the human control system for motion generation.

Please note that the biological information presented here is a precise but simplified model to be understood by the engineering community and of interest to this thesis.

3.1 Biological inspiration

The biological approach will be done within two ways: a neuroanatomical and functional way. Each description of every actor playing a role in

the human motion generation will include some details about its further possibilities under the scope of the proposed controller.

3.1.1 Introduction

We can immediately realize that each movement produced by a human involves many actors playing different roles. All of them are commanded by the central nervous system (CNS) which could be seen as the main character or controller of the motion [61, 62].

The CNS is an extremely complex element. The neuron is the elemental unit used to build the CNS and a healthy CNS contains around ten billion neurons.

Parallel to this concept, our system will be based on neurons as the fundamental block to develop the whole system; using either FPGA or VLSI chips.

The CNS includes the two important structures of the brain and the spinal cord [3]. Both of them are responsible for many important functions such as: walking, writing, running, etc.; almost any action we develop needs the CNS [63]. Among them, in the present dissertation we will focus our attention on motion control. If we make a classification of the movements conducted by a human, we can distinguish:

- Intended Movements: triggered by a stimulus or internal decision. All the action is done in a voluntary way. The more practise, the more rhythm they become.
- Rhythm movements: once again, they are triggered by a stimulus or internal decision. But in this case, they are voluntary, and occur only at the beginning and at the end of the movement. While the action is done, it is involuntary.
- Reflex Movements: they are triggered and are all done involuntarily.

When we refer to voluntary and involuntary, we mean that for some activities, maybe the ones learnt, only a little participation of the brain is necessary; we only realised for a while (usually at the beginning and ending), i.e. drive, swim, walk, run, etc. [64].

This thesis, specifically, is going to be focused on the intended movements to reach a target. This type of movement involves some agents which can be divided into three hierarchical and also parallel connected levels (Figure 3.2). The top level is for the motor areas of the cerebral cortex, the intermediate level is comprised of the subcortical structures such as the brain stem, the cerebellum, thalamic areas or the basal ganglia. The lower level is made of the spinal cord. Figure 3.2 shows this hierarchical structure and its components.

3.1.2 Functional motor areas of the cerebral cortex

There are five cortical areas involved in motor controlling (Figure 3.1): I and II primary somatosensory cortex areas and posterior parietal cortex (PPC) from the sensory cortical regions; primary motor cortex (M1), supplementary motor area (SMA) and premotor cortex (PM) from the motor cortical regions. The motor cortical regions are placed in the posterior region of the frontal lobe and the sensory ones are placed in the anterior region of the parietal lobe.

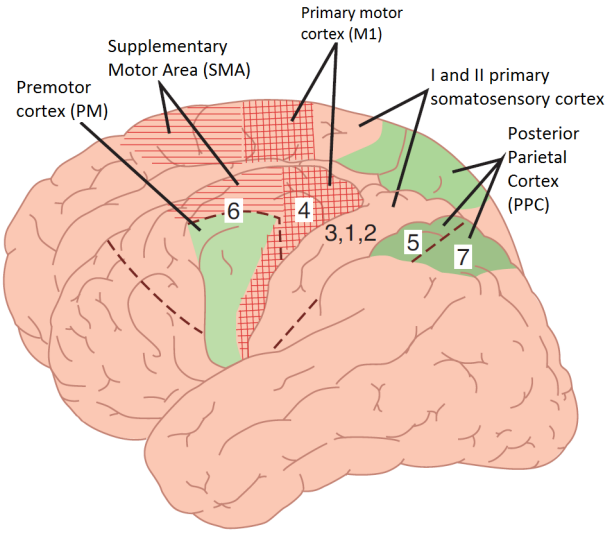

Figure 3.1: Cortical Areas involved in volitional movements. Taken from [3].

.

I and II Primary somatosensory cortex

This occupies the 3, 1, 2 Broadmann areas. It includes two somatosensory areas: I and II and it is located on the anterior parietal lobe. The function of these regions is to know the localization of the body and to get the position of the objects which can be interaction with by gathering the sensory inputs from receptors that are located along the entire body. Using specific receptors, the forms and features such as the weight, touch or texture of the objects can be estimated, too.

The somatosensory area I is in charge of the location of the thigh, thorax, neck, shoulder, hand, fingers of the hands, tongue and abdomen. On the other hand, the somatosensory area II is responsible for location of the leg, arm and face. However, to set up the II area some projections from I area are needed. These regions project to the posterior parietal cortex [3] sending the information obtained.

These areas are beyond the scope of this thesis. Thus, they will play the role of sensory information provider to the Posterior Parietal Cortex for our model.

Posterior parietal cortex

This occupies the Brodmann area 5 and 7 and it is located in the region previous to the somatosensory areas. It projects to motor cortical areas [4].

The main task of this area to develop the data received from the sensory areas to get further information about the localization and objects. Also, its tasks include the mapping of the frame of reference; i.e. it receives a combination of information from the sensory areas according to their frame of reference and PPC maps it into the proper frame of reference for motor commands [65]. In vivo experiments show that PPC could be also involved joined with other regions in corrective actions when a target is moved from its original position during a movement [66].

Thus, for the model proposed, this area could act as the mapper between the different frames of references.

Primary Motor Cortex

This occupies the Brodmann area 4. It is thought to have the main responsibility for volitional movements. Its tasks include coding the direction of the movement [67] and being involve in the final execution by the end effectors, the muscles. To accomplish with them, it receives projections from the premotor areas (probably including a kind of movement planning [62]) and also projections of all the other areas involved in the movement via thalamus. The primary projections reach the spinal cord in two ways: directly (mainly use for fine movements such as hands or fingers movements) and indirectly via the brain stem.

Although the primary motor cortex is considered the top level of the hierarchical structure associated to the motor control, some sensory responses can be observed inside it [68].

Since it is supposed to be the main controller of the whole movement, this will be the role in our controller.

Premotor cortex

This occupies the Brodmann area 6 and it is situated in the previous region of the M1. This includes the dorsal premotor cortex (PMd) and the ventral premotor cortex (PMv). Both are in charge of preparing the movement: first a motor virtual plan then a motor planning will occur. In this dissertation they will be treated as a unique area to avoid cluttering.

The premotor cortex is involved in reaching movements by sending projections to M1. As with the M1, the premotor cortex projects to the spinal cord in two ways: directly and indirectly via the brain stem.

Since the premotor cortex registers a short activation prior to the movement [69], it could be considered as the motion planner [62] by pre computing some details of the movement to be done. This will be the main task under the scope of our controller.

Supplementary motor Area

It is a small part of Brodmann area 6 too and it works together with the premotor cortex. It plays an important role in programming complex movement sequences not triggered by a stimulus. The activity of this area is highly correlated with the timing of movement sequences. It includes a preSMA area which is thought to be involved in reinforcement learning [70].

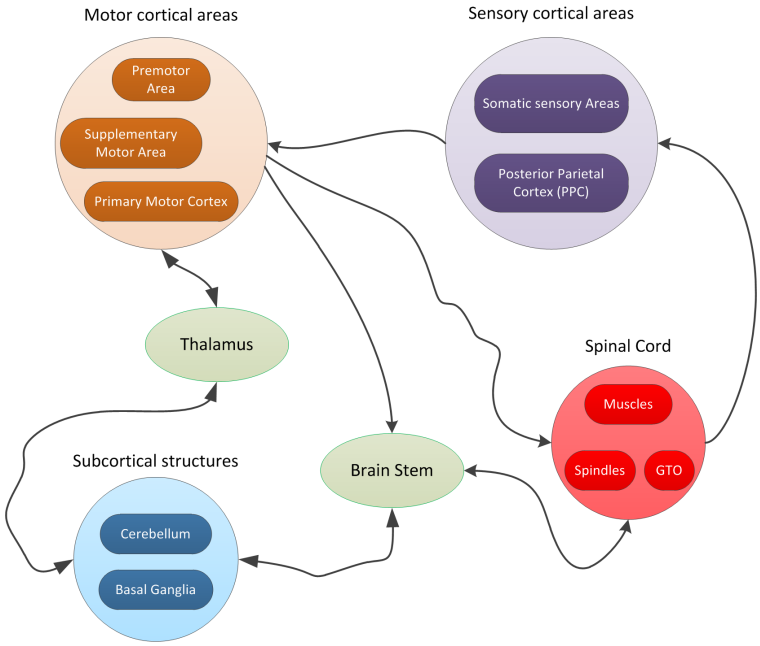

Figure 3.2: Hierarchical CNS levels and connections representation.

3.1.3 Subcortical Structures

Cerebellum

It is part of the encephalon and it is placed in the lower part of the brain. It is in charge of stabilizing the posture and correct movements in an indirect way. It fine tunes the projections coming from the descent motor systems

(corollary discharges). The cerebellum is able to adjust the projections because it receives them from all the hierarchical levels of the nervous system.

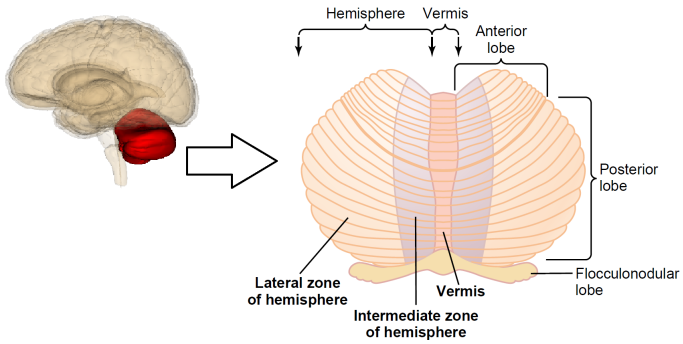

Figure 3.3: Left: Localization of the cerebellum in the brain. Right (taken from [4]): Functional parts of the cerebellum. The vestibulocerebellum comprised the flocculonodular lobe, the spinocerebellum the vermis and both of the intermediate zones of the hemisphere; finally, the cerebrocerebellum includes the two lateral zones of the hemisphere.

In principle, this tuned could be in the form of making a comparison between discharges and afferent information like a classical engineering feedback, but biological feedback loops are slow and have small gains. Some studies from Wolpert proposed that the cerebellum could allocate both forward or inverse models of the motor controller [71]. A forward model will predict the effect of a command set by higher instances. Then, the real feedback and the predicted one can be combined and used to correct the movement and also to keep updated the internal forward model. An inverse model can provide the neural command necessary to reach a trajectory. Thus, if a desired trajectory is supplied to the inverse model, it could generate the commands needed to achieve it. Then, the real commands and the ones generated by the model can be merged to minimize errors.

The cerebellum is divided into three functional levels coinciding with its anatomic divisions (Figure 3.3):

- Vestibulocerebellum: this is in charge of the equilibrium control and postural movements.
- Spinocerebellum: this receives projections from the motor cortex. Due to the information received, it can predict the future position of the limb. Thus, it can affect the ongoing movements.
- Cerebrocerebellum: this receives projections from cerebral cortex and it returns these projections by the thalamus. This level is involved in the planning and sequencing of complex movement.

The modelling of the cerebellum opens up exciting fields such as the learning methods to adjust connections.

In this thesis, the cerebellum plays the role of comparator between the order commanded (corollary discharge) and the feedback from the actuators. An error will result from this comparison. Then, it will be used to close the control loop and reach a more precise movement than if an open loop control technique were applied.

Basal Ganglia (BG)

The basal ganglia are the largest subcortical structure placed at the base of the forebrain (as a part of the encephalon). The main components are: the globus pallidus (GP), the striatum (caudate nucleus and putamen), the nucleus accumbens, the substantia nigra and subthalamic nucleus. The BG main tasks are to execute subconscious learned pattern movements and to execute cognitive planning of sequential and parallel motor patterns to achieve specific conscious goals. To accomplish with these tasks, they include some neuronal circuits such as the caudado and putamen.

The basal ganglia do not receive information from the sensory receptors. Instead, they receive projections from cortical motor areas and also from the amygdala. They do not have direct projections to the spinal cord, but via thalamus [72].

The scientific community accepts that BG play an important role in triggering and sequencing movements [3, 4, 73]. In [74] can be read: *"The basal*

ganglia plays an important, but enigmatic, role in motor control and learning, including reaching and pointing movements". Furthermore, the BG are involved in the reinforcement learning process of motor behaviour [75–78].

The experimental results presented in [79] strongly support the existence of movement gating performed by the BG. Nevertheless, the work presented in [4] lead to the hypothesis that the basal ganglia alone cannot account for full movement control and cortical structures must also be involved. According to [73], the initiation of muscular movements is preceded by activity in the premotor cortex followed by activation of the GP. In [80], the basal ganglia role in movement generation is explained as follows: once the cortex has come to a decision of making a specific movement (stimulus is delivered), the striatum (region of the basal ganglia target of cortical input) is activated. Then, internal complex activity of the BG associated to neurotransmitters releases the pathways to allow the movement. In particular, direct projections from the cortex to the striatum and subthalamic nucleus are believed to activate the basal ganglia [5]. According to [5], the input to the striatum is mediated by the N-methyl-D-aspartate (NMDA) receptors while the input to the subthalamic nucleus (STN) targets the non-NMDA channels. Furthermore, in [5], the authors support the idea that cortical activity is necessary to activate the BG.

Thus, they will play the role of gating and controlling the speed of the movement in our controller.

The Brain stem

This is a complex subcortical structure, placed in the posterior part of the brain. It includes the medulla oblongata (myelencephalon), pons (part of metencephalon), and midbrain (mesencephalon).

This is an extremely important part of the brain as the nerve connections of the motor and sensory systems from the main part of the brain to the rest of the body pass through the brain stem. This includes many sensitive and motor nuclei and tracts such as the corticospinal tract which is involved in volitional movements.

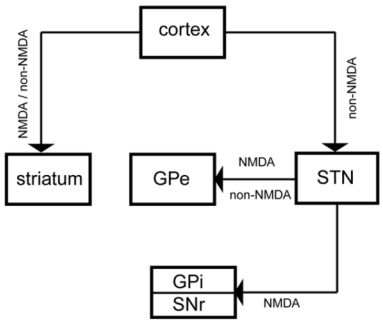

Figure 3.4: Block diagram of part of Basal Ganglia neural circuit to execute movements. It is not the whole system represented but further information would lead to include unnecessary data. GPe and GPi are the external and internal segments of the globus pallidus, STN is the subthalamic nucleus and SNr is the substantia nigra reticulata. Taken from [5].

Although it has other tasks, for our model it will be the gateway between *the execution elements* and the controller.

The Thalamus

This is situated between the cerebral cortex and the midbrain. This is also a very important structure. Its function includes relay the information between cortical and subcortical areas. It will be also the same in our controller: gateway between several controller parts.

3.1.4 Spinal cord

This is the lower element of the hierarchical motor control system. It gathers many neural circuits in order to facilitate many rhythm and reflex movements [4].

Apart from these neural circuits, and of interest to this dissertation, the spinal cord includes the cellular bodies of the motoneurons, interneurons and Renshaw cells. All of these cells together with the muscles are involved in the final motion execution [4].

The mechanism to activate the motoneurons follows the Henneman size principle [81]. To understand the Henneman principle or the size principle let us start by defining the concept of the motor unit. A motor unit consists of an alpha motoneuron and the muscle fibres innervated by its axon. One motoneuron can innervate several muscle fibres. Therefore, when a motoneuron fires a spike it goes to all of the innervated muscle fibres and they will contract together.

Hennemans principle or size principle [81] states that the recruitment of the motoneurons (or motor units) is done according to their size. The first motor units recruited are the smaller with a lower threshold and then the larger ones with higher threshold. Also, the force produced by these groups depends on their size. The smaller motor units consist of small muscle fibres that are fatigue-resistant but they produce smaller forces. In front of them, larger motor units produce higher peak forces but their muscle fibres are less resistant to fatigue [73].

Many biological studies confirm this principle [82–84], but not all of them; in [85], they said that the recruitment of them is done attending to their input resistances (IR). An in [86], the authors attempt to apply the size principle in the recruitment of motor units across muscles to reach smoother body movements.

Nevertheless, according to the original principle, we could say that, though the alpha motoneurons are all the same, they have different axon sizes.

Some observations related to this principle:

- There is a correlation between the discharges to the motoneurons and the force produced. The higher the discharges, the higher the force developed. The amount of force developed depends on both the firing rate and the motor units recruited.

- The central nervous system has two different methods to increase the force developed: recruit larger motor units or increase the firing rate of the recruited ones. The second method will provoke saturation on the force developed.

3.1.5 Muscles

This dissertation is focused on the muscles involved in movement: the skeletal muscles. These kinds of muscles are made of a combination of muscular fibres such as the extrafusal and intrafusal fibres. The muscles fibres are innervated by two different kinds of motoneurons: alpha and gamma. Each of them are responsible for innervate different groups of fibres.

The extrafusal fibres are in charge of contracting and stretching the muscles and they are innervated by the alpha motoneurons. Finally, the intrafusal fibres are in charge of the spindles and they are innervated by the gamma motoneurons.

Up to this point, we have defined how to activate an individual muscle. However, in any movement many muscles are involved; or at least two of them: agonist and antagonist. If we look at the forearm movement, we have two different behaviours: a contraction of the biceps muscle if we raise the forearm and a stretch if we move it down. To achieve an accurate forearm movement and avoid lateral motoneuron activations (which would provoke malfunction), there are some protective neuronal circuits. These are based on the Renshaw cells at a local level and on the interneurons (Ia) at an upper level.

The connection including the Renshaw cells, the Interneurons, and the motoneurons of both agonist and antagonist muscles is a mechanism to compensate the positional distortions when the objective is to increase the stiffness of the joint. The structure is shown in Figure 3.7.

3.1.6 Proprioceptive information

The CNS needs a feedback mechanism to know the current position of its effectors: the muscles and the limbs. The feedback is provided by two receptors: the spindles and the Golgi tendon organs (GTO).

These two organs, spindles and GTO, together are known as the proprioceptive feedback. This concept is related to the internal information of the joints.

Spindles

This are located in the intrafusal muscle fibres and they respond to both muscle length and velocity of the movement. Related to its morphology, each spindle is between 3 and 10 mm long and is made of 3 to 12 intrafusal muscle fibres.

These spindles have two types of sensory endings:

- Primary endings: they are innervated by afferent fibers that belong to group Ia. These neurons transmit the information to the spinal cord with a rate of 70 to 120 m/s [4].
- Secondary endings: they are innervated by afferent fibers that belong to group II.

The response of the spindles varies according to the kind of movement. If a slow stretching is taking place, both endings raise their firing rate and keep the rate even some minutes if and when the spindle remains stretched. Nevertheless, if a jerk stretching is taking place, only the primary ending raise its firing rate and only during the stretching. Nothing will happen in the secondary ending. When this jerk stretching is finished, the primary ending will return to its baseline firing rate [4].

One important point of the spindles is their location: inside, and in parallel with the intrafusal muscle fibres. These fibres are innervated by the gamma motoneurons. If these fibres are stretched and then they return to their idle state, also the spindles should shrink their size to avoid a malfunction. Thus, the spindles must be tune by the gamma motoneurons. The gain of the spindles depends on the activity of the gamma motoneurons.

As a result of this concept, a co-activation mechanism occurs to activate both gamma and alpha motoneurons to avoid errors at the measurement of the spindles.

Golgi tendon organs (GTO)

They are located where the skeletal muscles are inserted in the tendons. They are connected to between 10 and 15 muscle fibres and they can measure the tension of these groups of fibres to which they are connected [3].

The GTO are serially connected with the muscle fibres so they respond as much to the active as to the passive tension changes. The response will be a firing rate in proportion to the tension. This rate will increase the base line rate. Once the tension has finished, the base line rate will remain present. These rates are transmitted using fast fibres to local circuits of the spinal cord and eventually to the cerebral cortex. Thus, they keep the CNS informed at all times of the muscle tension.

3.2 Setting up a bio inspired controller

3.2.1 Introduction

There are many researches working towards brain-process modeling including, but we are going to focus our attention on those who have put their effort on the development of algorithms that describe the intended movements. First, we should define what the important points for a control algorithm are.

3.2.2 What are the important points? / Relevant points for motion controlling

We can distinguish between the issues relatives to control theory, which can be called the modelling issues, and going further, the important points to develop such model: the implementation possibilities.

On the one hand, if we want to create a control model for reaching tasks, the first option to considerer if the control technique applied will be an open or closed loop one. Thus, this is the first relevance point: the control technique. Once it is fixed, if a close loop is used, the feedback comes up. What are the sources? For human motion controlling, two possibilities are available: vision, and proprioceptive information coming from the spindles or GTO, respectively.

This feedback information coming from two kinds of sources may have different format and delays to deal with. Thus, here we have the second relevance point: the feedback and all related to its format and delay.

Once the feedback is going to be included in the model, the frame of reference to which it is referred have to be defined. This is one of the most important points.

Two frames of references are presented in human motion: the one of the vision system and the one understood by the effectors. They are called the vision and motor frames of reference, respectively. For every action we develop, a translation should be done if not all the magnitudes are referred to the same frame of reference. Even the proprioceptive information has to be mapped into the frame that it is useful for the motor control system. Part of this translating task is done by the PPC (see section 3.1.2).

This concept of frame of reference leads us to the fourth relevant item: which magnitude do we want to control. Traditionally, the position and speed are the ones controlled in reaching movements. However, the human CNS has information related to the length and tension (proprioceptive), and position and speed information coming from the visual feedback, too. Thus, they all are the magnitudes what can be controlled.

Concerning the deep of the model, we can go through a discussion of how many details should be included. A too detailed description could provoke a lost on the effectiveness and an over simplification could led to avoid vital concepts.

On the other hand, further possibilities such us simulation, implementation and testing are considered. When a neuro-inspired controller is modeled, though sometimes it is not possible, the final objective is to implement it. Thus, the first step to take is to simulate it to check the performance. There are a few simulators available such as Nengo [87], Brian [88], Neuron [89] ([90] review the tools available for spiking neural networks simulation). Once the results from the simulation accomplish the expected behaviour, it is time to go into the real hardware to run the controller. Here, the important points are the platform which will allocate the controller. Currently, as we have described in Chapter 2, some neuromorphic platforms are available; the number of neurons, synapses and interfaces to other devices are important points here. Eventually, the robotic platform to be controlled is at the end of the architecture. It could include 'N' degrees of freedom (DoF) or joints with the same number of motor. Nowadays, the possibility to have

muscles into the robotic platform is coming into reality. In any case, an important issue to go through is the interface to the robotic platform. How are the motors driven?

There are two different possibilities to drive the motors. The broadly use Pulse Width Modulation (PWM) where servomotors are used and the Pulse Frequency Modulation (PFM) where DC motors are used. The first one uses the width of the pulse to codify the information and the second one uses the frequency to codify the information. Appendix A shows a report of both modulations.

3.2.3 Motor control theories/models available

Focusing on human intended movements, the models available can divided into those who are static models, i.e. only the trajectory is generated according to a reference supplied and the dynamic ones, i.e. the forces necessaries to move the arm to the objective are stated. Both models have to play together if the human motion aims to be recreated.

The main models available in previous works are:

- The Vector Integration to Endpoint (VITE): it generates the trajectory to reach a position set as the target. The frame of reference of both the target and the trajectory have to be the same; either visual or motor [6].
- The Adaptive VITE (AVITE): it generates a trajectory, too. But, the target position supplied is referred to the visual frame of reference [91].
- Direction to Rotation Effector Control of Trajectories (DIRECT): also generates a trajectory. But now, both, target and motor commands are referred to the visual frame of reference [92].
- The Cerebellar Model Articulation Controller (CMAC): input commands and feedback are combined to create a memory address. This address stores the output commands [93].
- M. Hersh, in [94] proposed a controller which combines two VITE-like sub-controllers active in different spaces (joint angle and end-effector location spaces).

- Todorov and Jordan in [95] and [96] present an alternative theory based on stochastic optimal feedback control. They try to use the feedback in a smarter way where only the errors comprising the task are corrected.

The one selected to implement the trajectory controller is the VITE. However, a translation must be done prior to use it under the spike-based processing paradigm.

Once the trajectory is generated, it can be supplied to the end effectors to follow it and reach the target. At the same time, there is a control layer where the forces to reach each position are generated and controlled. The algorithm selected for this purpose is:

- Factorization of Length and Tension (FLETE): This algorithm maintains the position reached within different levels of stiffness by factorizing the length and tension of the muscles [6].

3.3 The VITE Algorithm

3.3.1 Introduction

In the late eighties, Bullock and Grossberg proposed a biologically realistic model of planned arm movements [72] broadly used at the robotics field [53, 94, 97, 98]. The model follows the principles of how intended movements are carried out by humans.

The VITE algorithm [6] tries to model the human movements keeping as many details of the neural system in mind as possible.

As it was conceived, VITE generates the trajectory to be followed by a joint. VITE does not compute any of the intermediate positions of each joint but the end effector.

VITE is the first layer involved in a planned arm movement. It does not integrate any feedback from the end robot. It generates the trajectory regardless of the forces needed to develop the movement. Thus, it feeds a second layer commanded by another algorithm. This second layer will

interface with the end effector and manages the commands received by the previous one.

3.3.2 Block Diagram and Equations

The block diagram (see Figure 3.5) and the equations (Equation (3.1), (3.2)) are presented in this subsection in their simplest form for the algorithm. A target is supplied to the algorithm. The difference between this target and the current position of the end effector (called difference vector, DV) will be integrated at each time in order to update the present position (Equation (3.2)). But, it will not be updated until the GO signal has a non-null value (Equation (3.2)). Meanwhile (GO signal has a null value) the difference vector is pre-computed in order to be ready for the shoot in the control signal. This time is known as the motor priming. If at any time while the movement is being done the GO signal goes zero, the movement will be frozen in that position.

The target position can be updated during the movement. This change will cause just an update in the difference vector related to the new goal.

All the magnitudes pointed in this algorithm must be referred to the same frame. Therefore, if spatial positions are considered, the integration of the present position will be matched with the speed profile of the movement.

If a comparison between this algorithm and the classical control theories for industrial applications (Proportional, Integral and Derivative controllers) is made, this algorithm would result similar to classical integral controller due to the final integral component but it is not. This integral block plays the role of the end effector or robot to reach some feedback to supply the 'ideal' position reached. The special component, GO signal, carries out a pseudo proportional playing the role of a pseudo disturbance.

$$\frac{d}{dt}DV = \alpha \times (-DV + TP - PP) \tag{3.1}$$

$$\frac{d}{dt}PP = DV \times GO \tag{3.2}$$

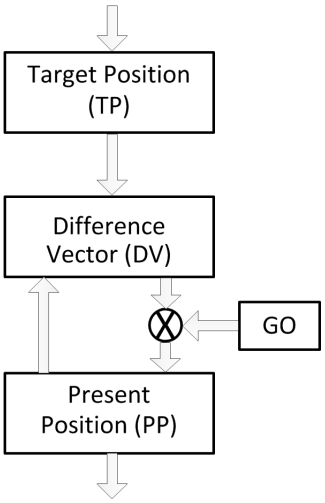

Figure 3.5: Block diagram. Redraw from [6].

Neurophysiological data supports the choice of the main computational blocks of the algorithm: in [72], some similarities with the population which codify the difference between the target position and the present position are shown. This DV population is matched with cells found in the precentral motor cortex (Area 4). In [72], also the gating mechanism involving a trigger signal is shown. The GO signal can be assimilated with the function done by the BG explained in section 3.1.3. The GO signal and the input coming from the DV both of them play the role of NMDA/non-NMDA connections previously described. Thus, the algorithm will not produce any output until the GO signal either target input have a nonzero value.

3.3.3 Some Considerations for the Algorithm Application

Synchronous Movement

With this algorithm, the movements are carried out in a real time and it is possible to change the target during the movement without disturbing it.

Thus, it receives an abstract reference, i.e., a spatial point and it should generate a trajectory for all the muscles involved in the action. In addition to this, the movement cannot be a composite of two or more joint movements (dotted lines in Figure 3.6; it must be a gesture or synchronous action of various muscles (solid lines in Figure 3.6). This concept is called synergies and they happen in a natural and dynamic way.

Thus, to perform a synchronous movement, each muscle group should contract or expand at a different quantity according to the difference vector computed for each one. From this concept of synchronous movement comes up the need for pattern and speed factorization. With an independent speed control for each muscle group it is possible to adjust all of them to reach high accuracy in the synchronization between all the muscles involved in the movement.

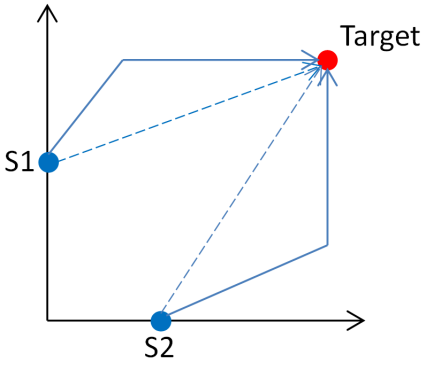

Figure 3.6: Two movements composed of two joints are represented. The start points are S1 and S2 and the target is the same for both. Dotted lines represent the right way to perform the movement and the solid lines indicate a composite of two actions.

End-effector Position

One important issue regarding this algorithm is how to know the position of an isolated muscle or a whole joint. From [72], two methods can be used

to know the position of the end muscle: the commands supplied (known as corollary discharges) and the inflow or proprioceptive information.

Therefore, if only the corollary discharges are used then it is supposed that the end effector arrives to the ordered position. In contrast, if the proprioceptive information is used, it is possible to update the position if a passive movement is performed and also to get a high level of accuracy in an intended movement.

The VITE algorithm uses only the corollary discharges to update the present position and therefore, to generate the trajectory. Thus, it is supposed that the end effector reaches the commanded position. This is a typical open loop control technique under classical approaches. However, regardless of whether the inflow information exists or not, it is necessary to implement gates to inhibit or allow the movement.

GO Signal and Speed Profile

The GO signal is in charge of the movement speed control. It is also the gate for that movement. The different this signal is, the different speed profile in the global movement achieved. The usual signal used is a ramp; the higher the slope, the faster the movement.

With a ramp profile for the GO signal, the general speed profile achieved is a bell-shaped one. The symmetry of this bell shaped profiles vary with speed [99]. Notice that this signal loses its meaning when the target is reached. To sum up, the worthy value of this algorithm is to reach a target, not how it is reached. So, the trajectory does not matter, except for fitting the joint angle constraints.

3.4 The FLETE Algorithm

This algorithm is in charge of the stiffness control. It operates in parallel with the VITE. It assures that any position reached by the joint can be hold with a range of stiffness without disturbing the position commanded as the target by higher layers.

The control is done considering three different populations of neurons: the motoneurons (alpha and gamma), the Renshaw cells and the interneurons per each muscle: agonist and antagonist (Figure 3.7 shows such elements).

In this algorithm, alpha motoneurons are responsible for innervate the muscles and provoke the movement. The gamma motoneurons are the gain controllers for the spindles. Both, the Renshaw cells and the Interneurons form two different neural circuits. The one close to the muscles is made of the Renshaw cells and the upper one is made of the interneurons. The main task of these circuits is to minimize the movements of the full joint by compensating the activity of each agonist and antagonist muscle [100] when a supplied target has been reached and the stiffness of the joint want to be modified.

Looking at the algorithm (Figure 3.7), A1 and A2 signals are the descending ones coming from the VITE algorithm. These signals will recruit a different number of motoneurons from each pool. It means that, according to the Henemman size principle, larger motoneurons threshold could have been reached on one of the motoneuron pools to achieve the position commanded. Then, if we supply a P signal to both pools, one of it will recruit larger motoneurons that will provoke an undesired movement in the joint. Eventually, these circuits are keeping away of an additional movement at the joint.

The spindles are responsible for providing the feedback to the alpha motoneurons and interneurons. The connection is an exciting one, thus, if the protection circuits are not well measured, the activity of the spindles can provoke a wrong behaviour of the algorithm.

The factorization of the length and tension is made using a signal (like the GO one) called P. This signal is sent in parallel to both pools of alpha motoneurons. The higher the value of this signal, the higher stiffness level achieved.

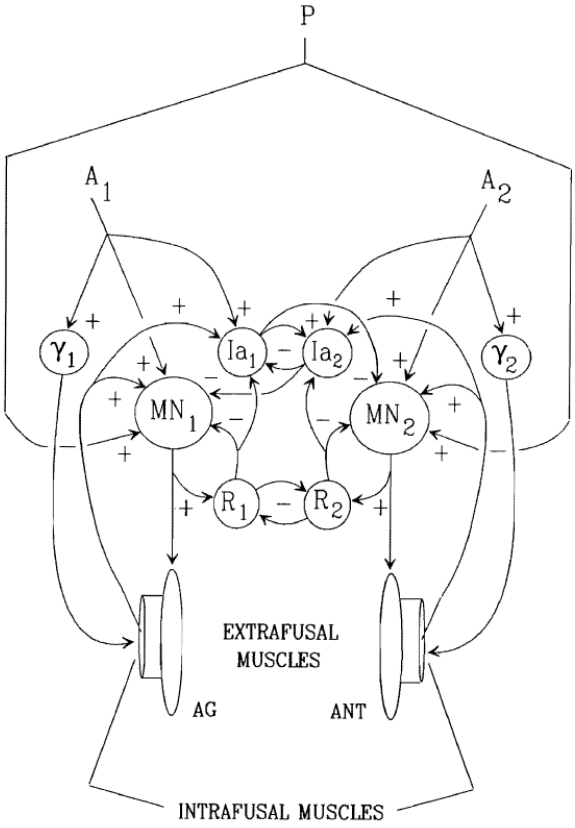

Figure 3.7: FLETE algorithm diagram. A1 and A2 signals are the descending signals coming from the VITE algorithm and P signal is the one used to control the stiffness of the joint. Ia are the interneuron populations, R are the Renshaw cells, MN the alpha motoneurons and γ the gamma motoneurons. Taken from [6].

Chapter 4

Event-based processing model

This chapter will focus on how the algorithms previously explained can be adapted for use with spiking neuron models. It includes a first part where the algorithms are prepared to be used within the spike paradigm. The event-processing blocks needed to implement the model are presented and explained. Then, a second part will describe the final model proposed to implement them, using the following platforms: FPGA and VLSI chips.

4.1 Introduction

This section presents a brief description of spike-based processing. This way of processing aims to mimic the behaviour of the human nervous system. The information in this system is analog and we try to reproduce it with digital (FPGA) and analog devices (VLSI chips). The design is made up of a hardware description language (HDL) of several blocks and a collection of python scripts to program the VLSI chips, respectively.

Here we have to make a distinction between digital and analog devices. The analog chips are using neuron models based on CMOS transistors [19]. Thus, the neuron features and the concept of analog information are inherent to them. In front of these chips, we have the digital device: a FPGA. Here

the behaviour of the neuron model is considerably different from the analog one and the information concept is slightly different.

In the case of digital device, the blocks process the information in the simplest way: addition (excitation), subtraction (inhibition) and injection (self-excitation) of spikes is allowed as this is supposed to be in a biological neuronal process. The information is based on the firing rate of the blocks trying to mimic the human neurons. On the other hand, the analog chips are designed to recreate some of the neuron features such as excitatory and inhibitory synapses, NMDA behaviour, short-time depression, etc. [19]. Thus, the range of operations available is very similar to that of a human being. Furthermore, in both cases there are only spikes flowing between these blocks, being processed while they flow, until they are directly applied to the motors.

On the digital device, the huge frequency of the clock is used to achieve equivalence to the way in which the information is processed within the analog devices.

We aim to reduce the complexity of using powerful processors. Also the power consumption is an important point in both cases: there is a huge difference of watts between the typical process computer and the electronic elements (two orders of magnitude lower usually). Another profitable advantage is space: we use small electronic devices which could be allocated in a stand-alone way.

References [41–43] present works where analog neuromorphic hardware and a FPGA are used and in which the power of this spike-based processing is revealed.

4.2 The Spike VITE algorithm (SVITE)

The VITE algorithm has been presented and thoroughly described (section 3.3). This section presents the translation of the algorithm into the spikes paradigm for use with FPGA and VLSI chips. We have called this new algorithm SVITE, for spike-based VITE.

4.2.1 FPGA implementation

Translation

To allocate the VITE algorithm into this platform and under the neuromorphic constraints, a translation has to be made. It is done in two ways: keeping the information in a spike-based system in mind (i.e. the firing rate) and taking advantage of the Laplace transformation to solve the equations considering zero initial conditions.

In these spike systems, the information has a relation with the inter spike interval (ISI) and specifically with the firing rate which can be understood as the frequency. That is the reason why we first go into frequency domain with Laplace (we are not matching Laplace domain with firing rate, it is just an interpretation to let us translate into spike-based processing paradigm).

Therefore, taking the equations of the algorithm as our starting-point and using the Laplace transformation to solve the equations, the main parts of the algorithm are translated regardless of whether the GO signal is used or not, because if we translate the equations in a strict way, the product between GO signal and the difference vector will be translated into a convolution in the frequency-domain and it will not be correct because this GO signal was designed in order to control the speed of the movement in the original algorithm. With this concept and regarding the information inside a spike system, the translation into spike paradigm for this product will be an addition of two spike trains in the spikes-processing paradigm. As a result, the firing rate (or frequency) of the resulting signal will be increased in any case. The next section deals with this idea. Thus, the translation starts by applying Laplace transform to equations 3.1 and 3.2. Then, equations 4.1 and 4.2 are obtained.

$$s \times DV(s) = \alpha \times (-DV(s) + TP(s) - PP(s)) \tag{4.1}$$

$$s \times PP(s) = DV(s) \tag{4.2}$$

$$DV(s) = \frac{\alpha}{1+\alpha} \times (TP(s) - PP(s)) \tag{4.3}$$

$$PP(s) = \frac{1}{s} \times DV(s) \tag{4.4}$$

By considering equations 4.3 and 4.4 it is easy to arrange them in blocks according to classical control theories [101] (Figure 4.1):

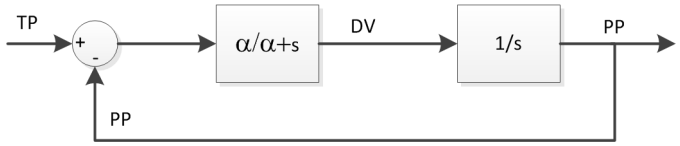

Figure 4.1: Block diagram resulted from the conversion of the VITE algorithm using the Laplace transformation.

Once we have the block diagram of the algorithm in the frequency domain, going through [102] and [103] and recovering the GO signal functionality, the resulting spike blocks are detailed in Figure 4.2.

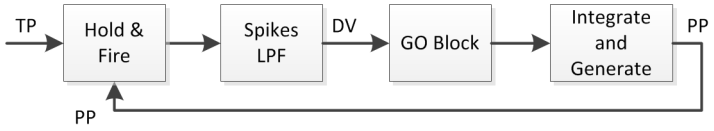

Figure 4.2: Block diagram generated from existing spikes processing blocks.

The blocks in use are described in [41] except for the GO block which will be described in detail in the next section [102].

GO Block

The main function of this block is to control the speed of movement and also to be the gate of it, but it has to deal with the fact that thinking in neuromorphic engineering, we do not have an element that carries out a multiplication as usual because it is not a biological function.

In the spike-based information codification, an approach to perform the GO function is to inject a determined number of spikes every time the previous block fires one, but equidistantly distributed over time as much as possible. It is like amplifying or exciting the activity, thinking in a biological way.

There are a few options to implement the block into the FPGA: to inject spikes according to the slope of a ramp, just one to N spikes per each received in a continuous way and so on. It can be implemented following any other function, but we have selected the ramp because it allows speed control and it is quite simple to implement inside the FPGA with counters. This selection allows us to configure the slope to achieve the desired speed. The final synthesized block is described in Figure 4.3.

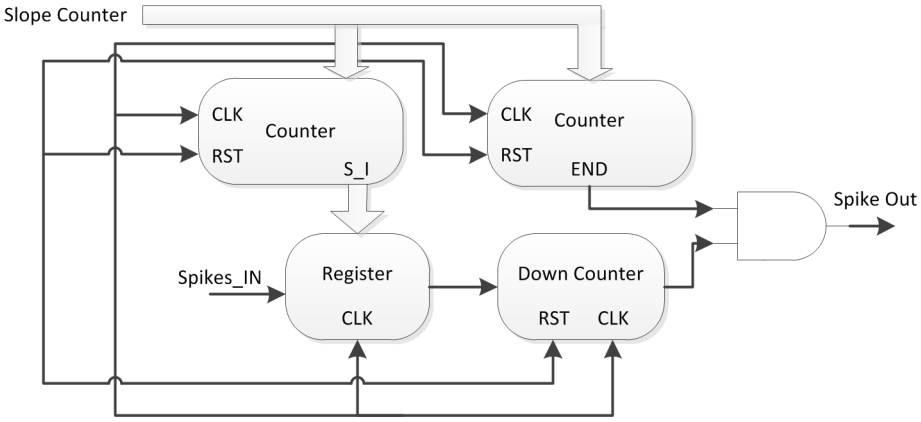

Figure 4.3: Block diagram generated for the block. It includes three counters: two of them are straight: one for the number of spikes to inject (S_I in the diagram) and another one for the life signal (it will produce the END signal to finish the movement); the last one is a decreasing counter in order to inject the accurate number of spikes.

The straight counters receive the slope counter parameter and produce the number of spikes according to the slope value of the ramp and the signal to finish the spike injection, respectively. Every time that a spike is received, the register value is updated with the number of spikes to inject. Figure 4.4 shows the behaviour explained.

If we design the block as it has been explained, the red thicker line behaviour in Figure 4.4 would be performed. It is a discrete result. A logical conclusion if we consider the spike systems: to inject or not a spike. Then,

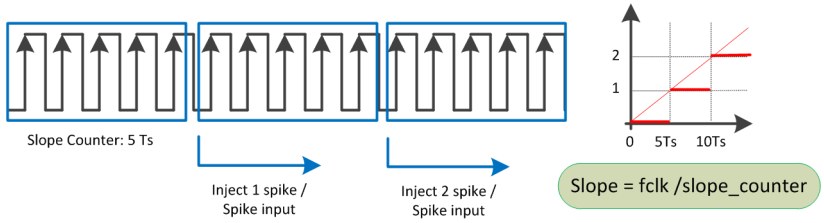

Figure 4.4: Explanation diagram of the implemented block. In this example, the slope counter is fixed to five clock periods; every time the count is reached one more spike will be injected. This way, and considering the firing rate, the discrete solid line is performed, and we were looking for the thinner line behaviour.

to reach the continuous solution (thinner line) it is necessary to include a low pass filter (spike-based and with single gain, too) at the output in the block diagram. This filter will distribute the spikes uniformly. As described in [41], and shown in Figure 4.5 each LPF block is composed by an I&G block with a feedback loop using another H&F block. In such a configuration, the Laplace transfer function of the system correspond to a low-pass-filter.
Thus, by including this filter a problem occurs: the I&G at its output. As it was explained, the I&G block keeps an n-bits count for the incoming spikes and generates spikes according to that count. So, we have to avoid the overflow of the count. Thus, we should finely tune it in order to avoid the saturation of the whole system.

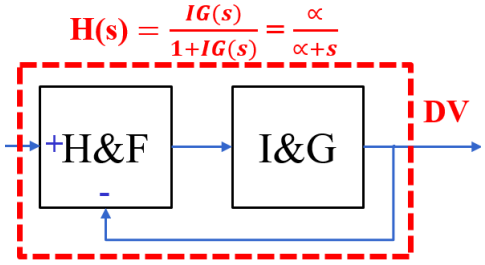

Figure 4.5: Spikes Low-Pass-Filter decomposition into I&G neuron and its Laplace Transfer function.

In any case, latency at the beginning will be calculated from the first count that injects any spike. During this period, the Difference Vector will be calculated by the previous part of the algorithm. Also, this time it is consistent with the fact that in a biological movement a previous activity is detected in the premotor cortex [69]. The latency is defined as follows:

$$latency = \frac{slope\_counter}{fclk} \tag{4.5}$$

Two important facts of this block are:

- It is important to saturate the slope achievable. Otherwise, the massive injection of spikes will saturate the complete system.
- The validity of this block is limited by time. It fits with the time-limited connection between the premotor cortex and the primary motor cortex [62], but it is necessary to fine-tune the limit in order to reach the target. We use this time limitation to consider the GO signal as a disturbance for calculating the system stability. Please refer to [8] for detailed descriptions of the stability analysis.

4.2.2 Final proposed SVITE model

FPGA implementation

The final neuro-controller proposed to implement on the FPGA is composed of four different types of spikes processing blocks [102, 103]:

- Hold and Fire (H&F): this block performs the addition or subtraction of spike flows to compute the error signal. This block has two inputs: one excitatory coming from the visual processing layer and one inhibitory from the end-block of the algorithm.
- Spikes low pass filter (LPF): the behaviour of the block is the same as an analog classical low pass filter but it operates with the spike's input firing rate. The result of this block is a uniform distribution of the spikes input [103]. There are two filter blocks in the SVITE algorithm: one at the H&F's output and another one included in the GO Block.
- GO Block: the main function of this block is to control the speed of the movement and also to be the gate of it. It is done by modifying

the input firing rate. We inject spikes according to a user parameter which define the speed desired.

- Integrate and Generate (I&G): this block includes one pre and post-synaptic connection and an integrator which computes the ongoing spikes. It models the activity level of the neuron; its firing rate depends on the integrated value. In its basic model, the integrated value is updated for each incoming spike by an increment or a decrement of the membrane potential (MP), modelled by a counter, depending of the polarity of the received spike. If the spike is positive MP is incremented, but it is decremented when the spike is negative. Current MP is continuously translated into an output stream of spikes.

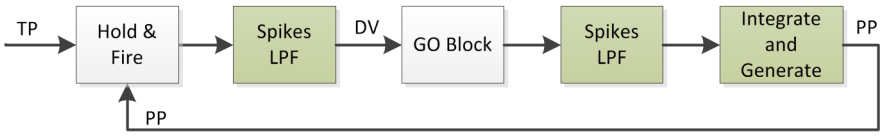

Figure 4.6: Block diagram of SVITE algorithm final model proposed.

This algorithm conforms something similar to a forward model and evaluates the corollary discharge with the Integrate and Generate block. So, no sensory discrepancies are noticed within this algorithm as it was expected without feedback from the robot. The assumption is that the commanded position is reached.

All of these blocks have a fully deterministic behaviour. In principle, for motor controlling it does not seem feasible to have random spikes distributions inside the controller. However, in this thesis we include a study where pseudo-random distributions of spikes have replaced deterministic ones in the motor controller (the blocks involved in this statistical study are the green ones at Figure 4.6). Statistical models will be presented at the results section (5.2.1).

This study is done because there are in-vivo experiments that show variability at the firing rate pattern if a constant stimulus is presented within

different tests at different times [104]. This behaviour is extremely problematic if the current trend is to match the neuron response per each stimulus presented to elaborate a map between stimuli and firing patterns. Regarding this view, [105] points out that if the stimulus presented is fluctuating, the neuron will produce a precise spike timing response.

For a deterministic spike distribution motor controller, if we consider this controller as an isolated part of the architecture, there would not be any noisy or random spikes. However, if we consider this controller as part of large neuromorphic architectures, we must face these different possible distributions of spikes and try to minimize their effects or at least predict the behaviour of the motor controller and act in accordance with them.

VLSI chip implementation

In this case, the translation is much easier than the previous one. We have predefined the chips to use [19]. They have the LIF neuron model as the basic element to build the architecture.

The spiking network defined for this implementation comprises three different populations: the DV population encodes the difference between the target and the estimated current position; the GO population implements movement gating and speed control; the PPC population encodes the estimated current position of the robotic platform. A schematic drawing of the proposed neural network is shown in Figure 4.7.

Four excitatory connections are part of the network: a couple of them are inputs and the other two are used to connect populations. The target stimulus excites the DV population and it is tuned to excite the membrane potential until produce a one to one relation between presynaptic and postsynaptic potentials. The GO signal stimulus excites to the GO population and it is a signal which increases over time. Short-time depression in the synaptic connection prevents the production of post-synaptic spikes in absence of a target stimulus. Finally, we have two excitatory recurrent synapses: one to connect the DV population with the GO population (as described in the next paragraph) and the other one to connect the GO population and the PPC population to update the position by integrating the incoming spikes.

The GO population has a special synapse that connects it to the DV population. This synapse plays the role of the NMDA channels described in section 3.1.3. With this synapse, the desired behaviour is achieved: neither the target nor the go stimulus presynaptic spikes in an isolated way will produce any postsynaptic action potential, which results in the desired gating movement function. The NMDA synaptic circuit produces an output current only if the membrane potential of the post-synpatic neuron is above a fixed threshold, referred as NMDA threshold and set by an input bias voltage. The excitatory synapse of the GO signal will keep the membrane potential higher than the NMDA threshold but without firing. Then, when a presynaptic spike occurs in the connection with the DV population, the membrane potential is higher than the NMDA threshold thus allowing the NMDA synapse to produce an output current and trigger a spike in the postsynaptic neuron.

The last connection is the inhibition between PPC and DV population. This synapse is tuned to inhibit the output firing rate of the DV population when the PPC has reached the firing rate set by the stimulus i.e. the robot has reached the target location.

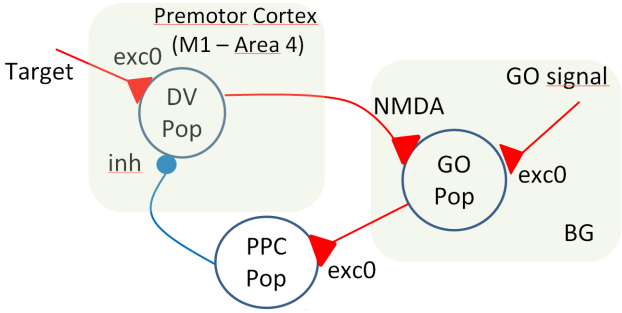

Figure 4.7: Spiking neural network diagram.

4.3 The Spike FLETE algorithm (SFLETE)

This algorithm considers well-known neuron populations, so no translation is necessary for it. Thus, to implement the algorithm we are going to use

the same populations and connections as they were shown in Figure 3.7 motoneurons, Renshaw cell and interneurons.

4.4 Considerations to apply to classic robotics platforms

We have maintained the original approach where the algorithms were thought to be used under the same constraints of human end-effectors: the muscles. However, this option is not yet available for the classical robotic platforms based on motors.

Thus, we have kept the model and simulate most of the parts but, eventually, the architecture implemented which run the robot is without the SFLETE algorithm. The reason is because using a robotic arm or either robotic platform, only a small number of joints are available and all of them are run by motors, not muscles. Furthermore, forces are not needed. The way to implement something similar to forces could be by measuring the current consumption by the motors; this current can be matched with the tension of the muscle and then this sensory source can replicate the GTO organs. In any case, the spindles cannot be replicated because the length of the joints is fixed. Instead of these sensory sources we are going to use motors that include encoder devices that allow measuring either the position or the speed.

4.5 Introducing the feedback: Final model

This section aims to merge all the models and algorithms already presented in a useful architecture that includes feedback. It is described in Figure 4.8.

The feedback block will play the role of the cerebellum (section 3.1.3). It receives the information related to the trajectory or speed profile from the SVITE and the sensory information available from the robot. By making a comparison between both information flows, the information supplied to the robot is tuned according to the desired movement. The addition of this new block enhances the model by closing the control loop.

The SFLETE algorithm will operate in parallel and it generates the forces to assure that the position commanded by SVITE is maintained. Also, it provides a stiffness controller to apply to the reached position by the joint. It needs the sensory information from the robotic platform and the information from the SVITE algorithm to generate the forces required by the muscles of the joint.

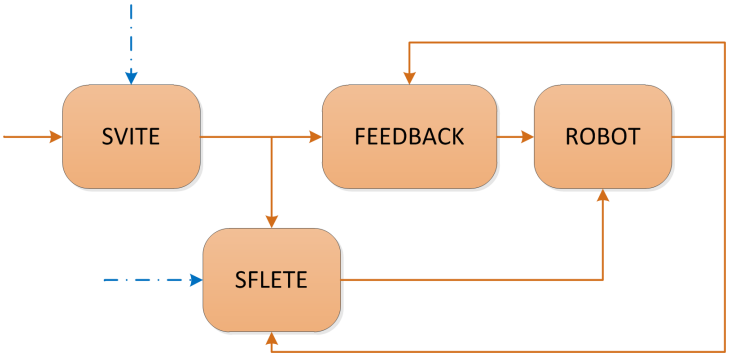

Figure 4.8: Full control architecture proposed for further research. The dotted blue lines represent the signals that can trigger the movement (in case of SVITE line) or the stiffness level (in case of SFLETE line).

Chapter 5

Results and Discussion

This section will present the results achieved by the research within this thesis. Two main sections are presented: simulated results and hardware results. Then, each of this section includes several subsections where the graphs are shown and explained. Finally, a discussion section is presented.

5.1 Simulation Results

This section includes the results achieved using three different software platforms:

- Simulink by Matlab plus Xilinx System Generator: this is used to simulate the behaviour of the blocks designed and eventually to check the function of the whole SVITE and SFLETE algorithm when they are thought to implement on a FPGA.

- Python and ipython framework: this is used to simulate the algorithm behaviour under the VLSI chip designing constrains.

- Statgraphics Centurion software package: all the statistical studies were conducted using it.

The advantages of using the simulator are: the tests can be arranged and repeated faster than using hardware, the price is usually lower than the

platform, and they are portable and flexible since the tests can be done anywhere. The disadvantages of the simulators are that they do not include all the functioning details of the hardware, so they cannot replicate it precisely. Besides, not all the cases can be simulated.

5.1.1 Software platform: Simulink by Matlab

These results have been achieved by means of these tools: MATLAB and Xilinx System Generator. The theoretical and simulation scenarios are described, then the simulated results are shown. The System Generator tool allows us to describe the blocks using VHDL. Thus, these results are obtained to check the behaviour of the FPGA designs. However, not all the cases to simulate can be carried out because of memory problems.

GO block results

The simulation setup used to check the behaviour of the GO block is the one shown in Figure 5.1.

Figure 5.1: Setup simulation scenario to check the GO Block behaviour.

The input to the Spikes Generator block is a constant value and it generates a firing rate according to the input value. The firing rates of the blocks follow these equations:

- Spikes Generator

$$f_{spikes} = \frac{f_{clk}}{2^{(NBITS-1)}} \times input \tag{5.1}$$

- GO Block

$$f_{spikes} = \frac{f_{clk}}{slope\_counter} \times input \times time \tag{5.2}$$

$$slope = \frac{f_{clk}}{slope\_counter} \tag{5.3}$$

- Spike-based Low Pass Filter (LPF)

$$Gain = 1 \tag{5.4}$$

$$\omega_{cut-off} = \frac{f_{clk}}{(2^{(NBITS-1)} \times (IG\_FD + 1)} \tag{5.5}$$

$$slope = \frac{f_{clk}}{slope\_counter} \tag{5.6}$$

The results are shown in Figures 5.2, 5.3, 5.4, 5.5, 5.6, 5.7, 5.8. Figure 5.2 shows how the LPF spike-based added at the end of the chain, uniformly distributes the spikes coming from the GO Block. Also, the latency expressed in Equation 4.5 can be seen at the beginning of the output spikes from the GO Block.

Figure 5.3 shows the effect produced by the LPF included at the end of the chain.

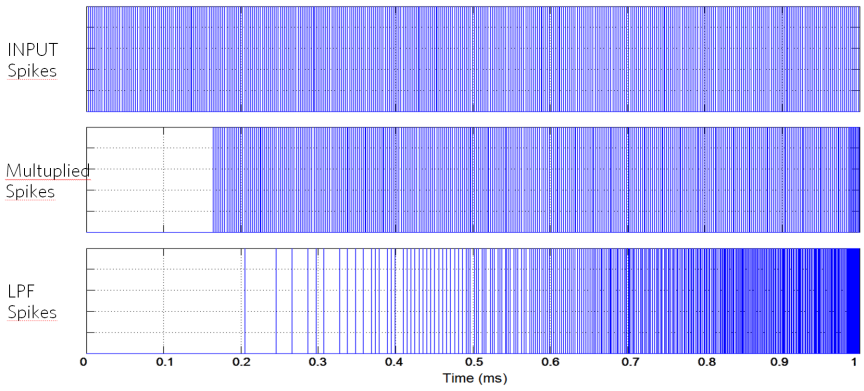

Figure 5.2: Spikes at the output of each block. There are three rows respectively showing: output of the spikes generator block, output of the block in which the spikes are injected and the output spikes at the LPF spike-based.

Figures 5.4 and 5.5 show the behaviour obtained when the slope_counter parameter changes its value.

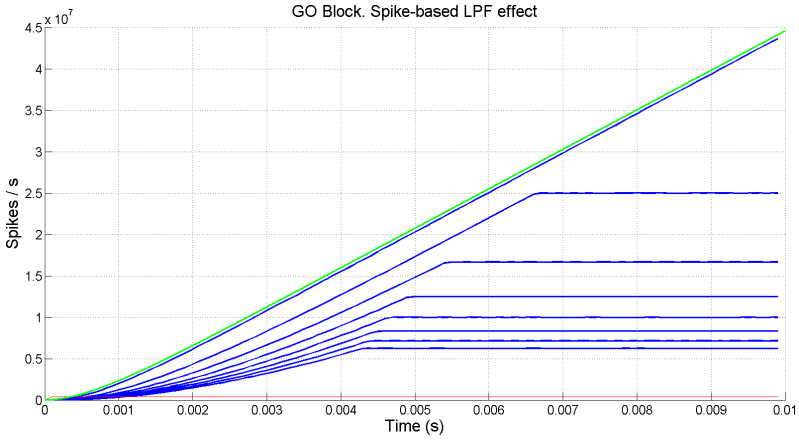

Figure 5.3: Output firing rates when an input rate of 390,625 Kevents/s is supplied by the spikes generator and a slope_counter parameter value of 2^{12} (i.e. slope of 12,207.03 %). The IG_FD parameter of the filter is changed from 0 to 7 (slowing down the cut off frequency of the filter). Green lines represent the behaviour according to the original algorithm and blue lines the one according to the block designed.

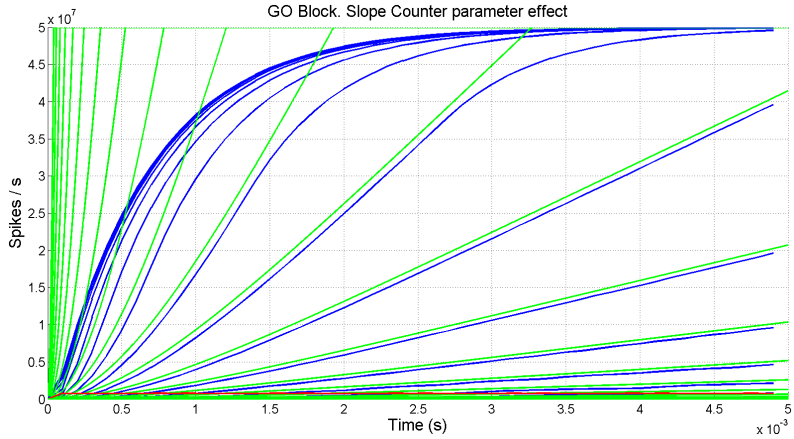

Figure 5.4: Output firing rates when an input rate of 390,625 Kevents/s is supplied by the spikes generator and a slope_counter parameter changing its value from 2^{10} to 2^{23}. The IG_FD parameter has a zero value and the I&G included with the filter is implemented with 16 bits. Green lines represent the behaviour according to the original algorithm and blue lines the one according to the block designed.

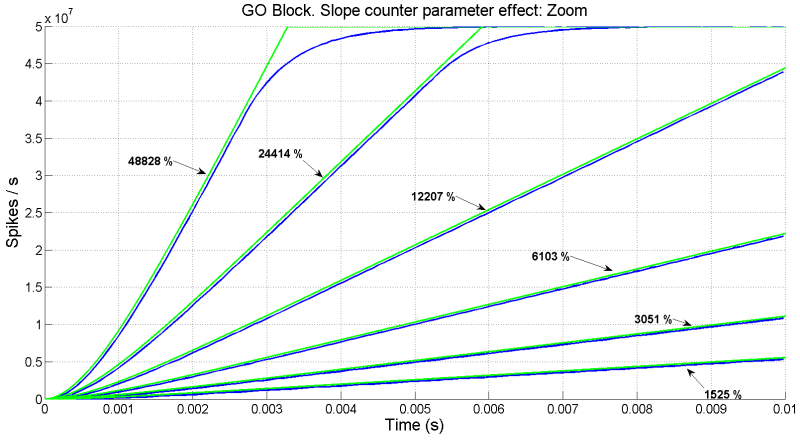

Figure 5.5: Zooming of figure 5.4 for a slope_counter parameter from 2^{10} to 2^{15}.

Table 5.1: Relationship between the time to saturation and the permanent regimen error when the I&G in the spike-based LPF is implemented with different number of bits.

Number of bits (I&G of LPF)	**Time to saturate (s)**	**Permanent reg. error**
6	2.5×10^{-5}	1,5 Mevents/s
8	2×10^{-5}	0,4 Mevents/s
10	7×10^{-5}	0,1 Mevents/s
12	2.6×10^{-4}	Aprox. 0
14	8.6×10^{-4}	Aprox. 0
16	3.2×10^{-3}	Aprox. 0
18	$> 0,01$	Aprox. 0

Analyzing Figure 5.4, the incorrect following of the input for high inputs (minimum slope_counter) appears. This is due to the fact that I&G included in the LPF goes into saturation region.

When the setup is put under saturation conditions, i.e. maximum input value: $2^{16} - 1$, slope_counter parameter equal to one and IG_FD parameter equal to zero, figures 5.6 and 5.7 show the behavior of the output firing rate. Table 5.1 summarizes the data obtained.

However, these results are not real because of the limits of the simulation environment; we have to select very high values like 12207,03125 % (212 value for the slope_counter parameter) for the slope to avoid memory errors. These values are not usable within a robotic platform.

Away from these saturation conditions, figure 5.8 shows the behaviour performed by the block.

The main conclusion obtained from the simulations is that there is a high dependency between the speed response of the system and the number of bits used to implement the I&G included in the spike-based LPF. As the

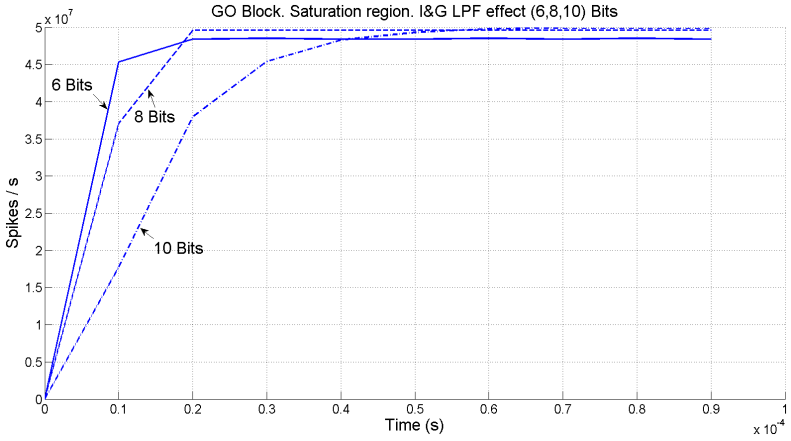

Figure 5.6: Output firing rate under saturation conditions when the I&G is implemented with 6, 8 and 10 bits.

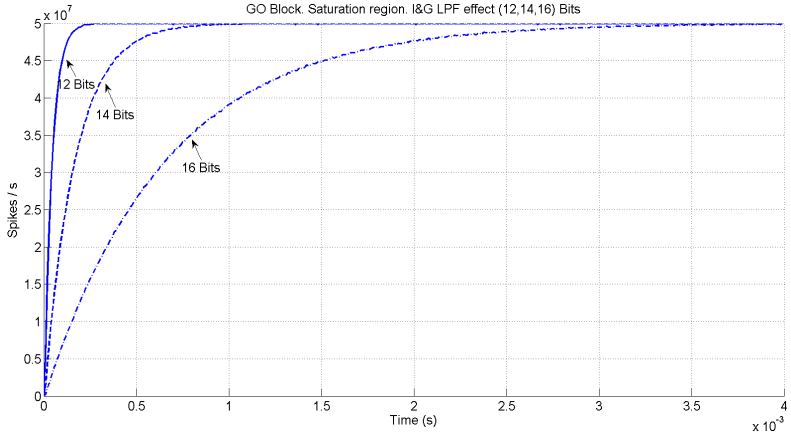

Figure 5.7: Output firing rate under saturation conditions when the I&G is implemented with 12, 14 and 16 bits.

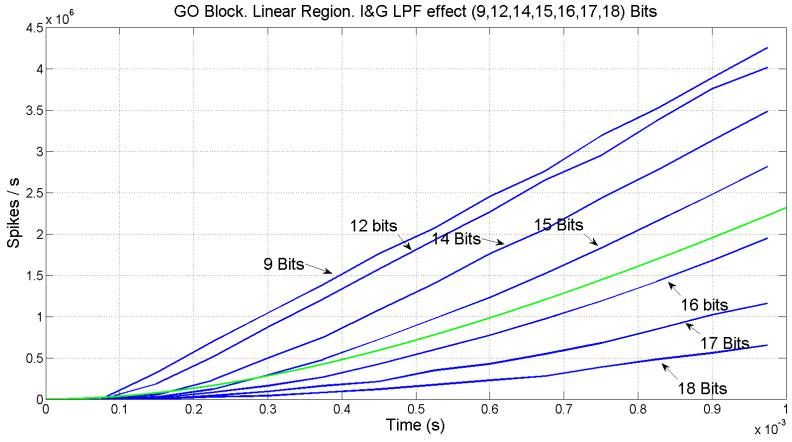

Figure 5.8: Output firing rates when an input rate of 390,625 Kevents/s is supplied by the spikes generator and a slope_counter parameter value of 4,096 (12,207 % slope). The I&G is implemented with 9, 12, 14, 15, 16, 17 and 18 bits. The green line represents the behaviour according to the original algorithm and blue lines the one according to the block designed and the parameter shown.

plots show, for very high values at the slope, 9 bits to implement the I&G could be a good option in front of 16 bits for more usable values at the slope.

The hardware results section will show more accurate results because of the simulation environment limitations. The Simulink from Matlab® plus System Generator by Xilinx® only allows simulations up to 10 ms.

SFLETE results

The scenario proposed to simulate the SFLETE algorithm is shown in Figure 5.9. Each population of neurons (Renshaw, Interneuron and motoneurons) is modeled with an I&G neuron model block. Each pool of motoneurons receives inputs from a block that mimic the SVITE behavior in order to imitate the original algorithm.

Before simulating all the algorithm elements, the first step to take is to characterize the lower protection neuronal circuit. This is the excitatory-inhibitory connection between a pool of Renshaw cells and a pool of motoneurons. Figures 5.10 and 5.11 show the peak firing rate and the times span of the motoneuron activity, respectively. These two graphs allow us make a good decision regarding the number of bits to perform the final implementation of the pools of motoneurons, renshaw and interneuron cells. The higher number of bits used, the higher peak rate and time length reached. So, a medium value to select is 10 or 12 bits.

The simulations are done by manually setting an activation of the spindles. There are two blocks to translate a constant input parameter into a firing rate. This firing rate will simulate the response of the spindles. They will project to each pool of motoneurons and Interneurons. Thus, the activity of the spindles are conditioning the interneurons design.

Figures 5.12 and 5.13 show how, under different inputs from the SVITE to the motoneuron pools (but the same for both plots) and the appropriate response of the spindles (higher on the side of the higher input rate from SVITE), the higher the number of bits used to implement the Interneuron, the smaller the response obtained from the motoneurons. This behaviour cannot be improved when both pools of interneurons are implemented with the same number of bits. Besides, as Figure 5.14 shows, when the interneuron

pools are implemented with a smaller number of bits than the motor or the Renshaw neuron pools, the behavior is not improvable anymore. The minimum activity from the motorneurons is obtained when both interneurons are well measured regarding the spindles activity besides they are both implemented with an equal or higher number of bits of the motor and Renshaw neuron pools. Besides, Figure 5.15 shows how a non-accurate (extremely high) response of the spindle can modified the response of the motoneurons by provoking a non-desired activation. This is because the spindles excite the interneurons and these do with the motoneurons. But, if we make the response of the motoneurons higher by using more bits to implement them, the effect of a huge response from the spindles could be mitigated as is shown in figure 5.16. This is because the motoneurons inhibited the response of the interneurons which are excited by the spindles.

Furthermore, this scenario offers the possibility to simulate the reflex movement mechanism. Figure 5.17 shows how, with all the inputs disable, simulating a short activation at the spindles, the motoneurons of both muscles response producing a reflex movement. The response of the motoneurons is inhibited by the Renshaw and Interneuron neuronal circuits.

5.1.2 Software platform: Python

SVITE results

This section presents the results obtained using the Brian simulator to verify the performance of the network described in the section 4.2.2. The simulations are done with one neuron per each population (one-to-one fashion connected). Figures 5.18 and 5.19 show the firing rate of each population including both stimuli: the target which excites the NMDA and the GO signal which excites the non-NMDA receptors. Figures 5.18 and 5.19 show how an initial non-zero GO signal in coincidence with a delayed target stimulus does not trigger any response, which is only present when both signals are active. When the GO stimulus set the membrane potential of the GO population higher than the NMDA threshold, the arrival of a presynaptic spike from the target will provoke a post-synaptic spike. To get this behavior related to non-NMDA channels and prevent the firing, the excitation connection of the GO stimulus is the one which includes a short time depression mechanism; otherwise, the time increasing firing rate of the GO stimulus will make the neuron fire. Figure 5.19 shows how increasing the slope of the

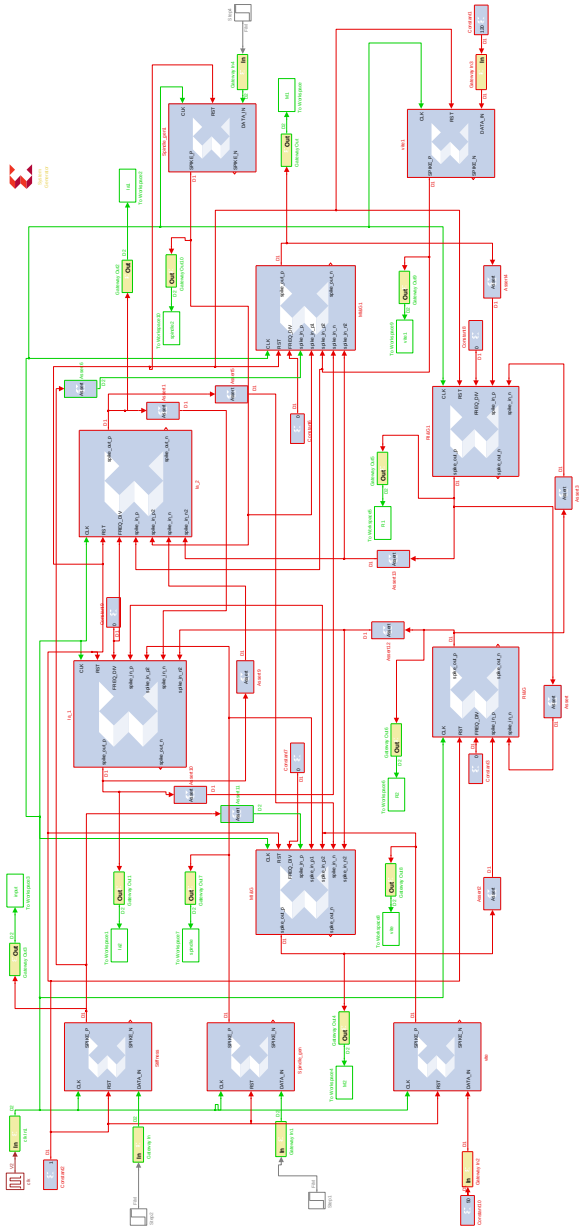

Figure 5.9: SFLETE algorithm scenario sat up to carry out the tests. The System Generator by Xilinx blocks can be seen in this model.

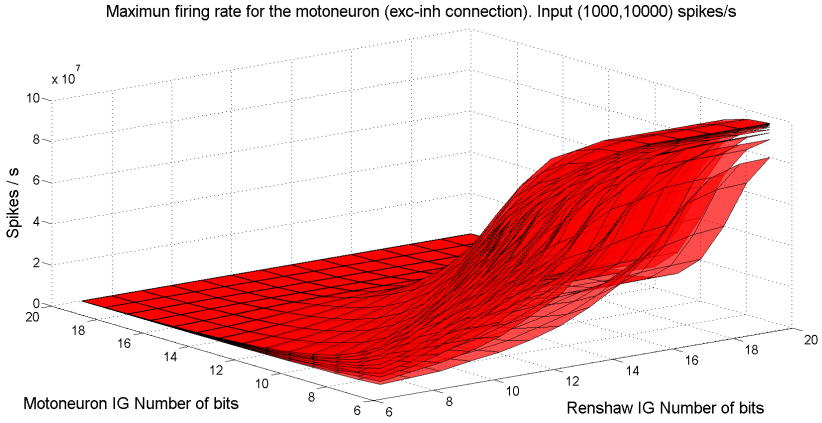

Figure 5.10: Firing rate peaks of the motoneuron pool when the Renshaw cell and the motoneurons are both implemented within a range of (7-20) bits. The input goes from 1 Kevent/s to 10 Kevents/s.

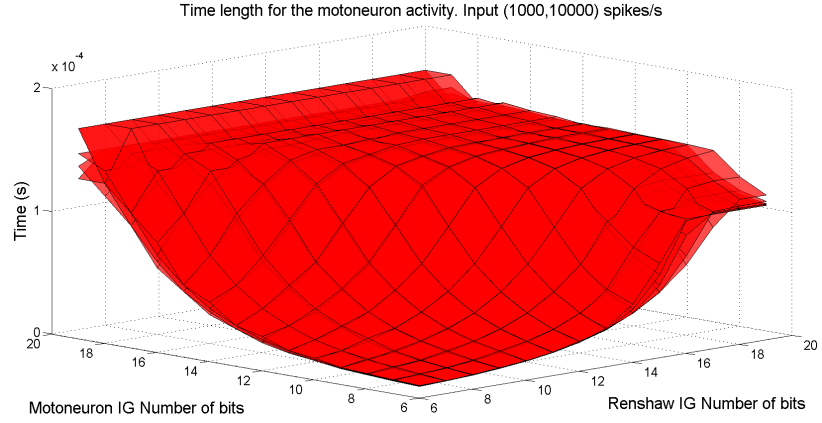

Figure 5.11: Motoneuron pool time length activity when the Renshaw cell and the motoneurons are both implemented within a range of (7-20) bits. The input goes from 1 Kevent/s to 10 Kevents/s.

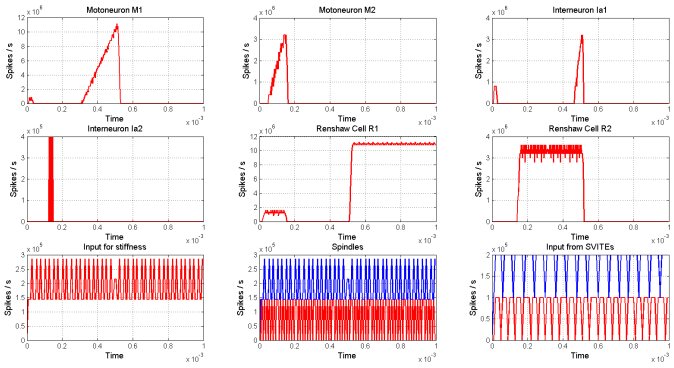

Figure 5.12: Firing rates of each population. The input to control the stiffness is fixed at 1,83 Kevents/s. The simulated input coming from SVITE is different per each muscle (red and blue graphs in the last subplot): 1,83 Kevents/s and 0.76 Kevents/s (this is mimicking a permanent regimen at the SVITE). The output set for the spindles is 183.1 Kevents/s and 76.29 Kevents/s for each side. The motoneurons and Renshaw cells are implemented with 10 bits and the Interneurons with 12 and 9 bits respectively.

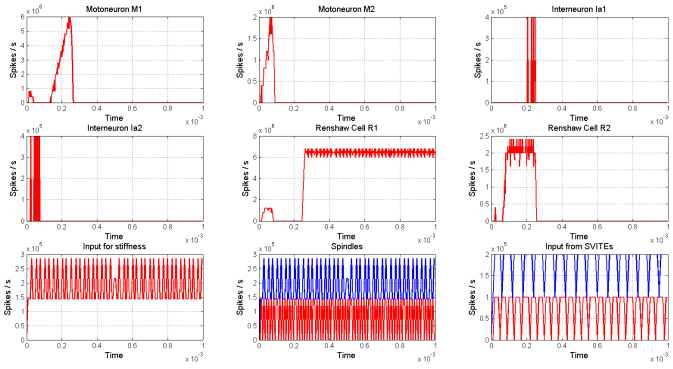

Figure 5.13: Firing rates of each population. The input to control the stiffness is fixed at 183 Kevents/s. The simulated input coming from SVITE is different per each muscle (red and blue graphs in the last subplot): 183 Kevents/s and 76.2 Kevents/s (this is mimicking a permanent regimen at the SVITE). The output set for the spindles is 183.1 Kevents/s and 76.29 Kevents/s for each side. The motoneurons and Renshaw cells are implemented with 10 bits and the Interneurons with 12 and 9 bits respectively.

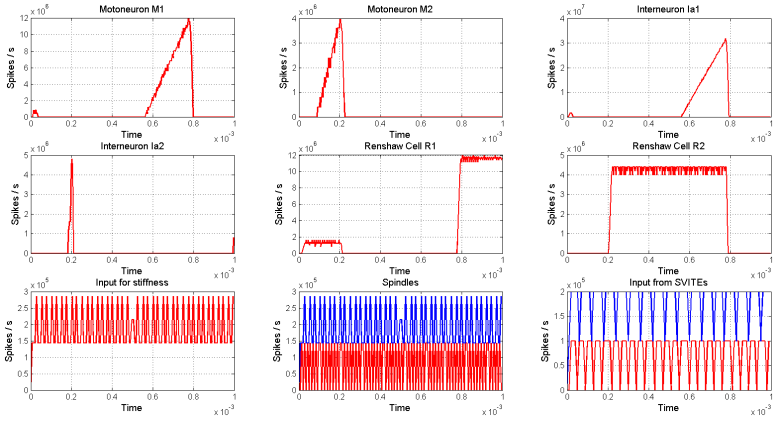

Figure 5.14: Firing rates of each population. Same conditions of Figure 5.13 but, in this plot, the Interneurons are implemented with 6 and 8 bits respectively, i.e. a number of bits smaller than the motor and Renshaw cells.

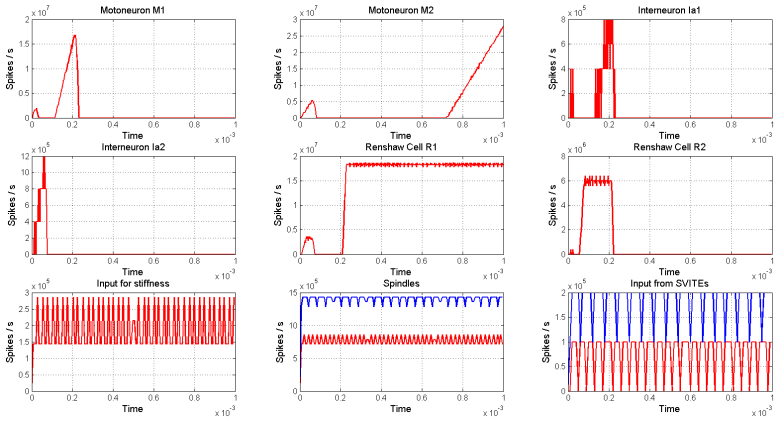

Figure 5.15: Firing rates of each population. Same conditions of Figure 5.13 but, in this plot, the spindle response is higher, up to 1.44 Mevents/s and 762 Kevents/s on the other side.

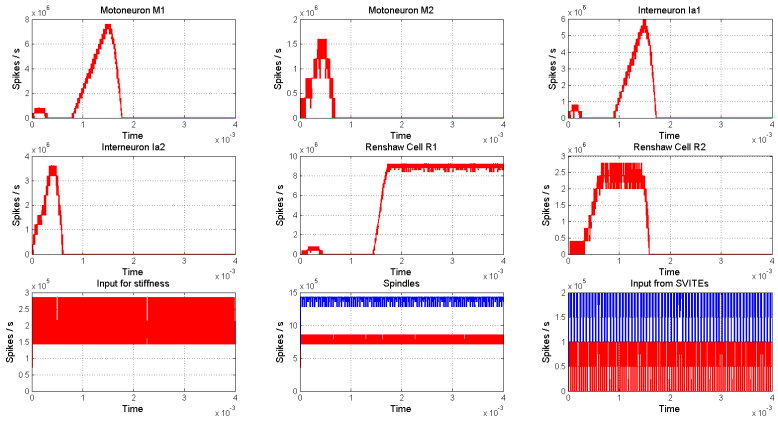

Figure 5.16: Firing rates of each population. Same conditions of Figure 5.13 but, in this plot, the motoneurons and Renshaw cells are implemented using 14 bits. This is the low bit number to use if the wrong behaviour wanted to be avioded.

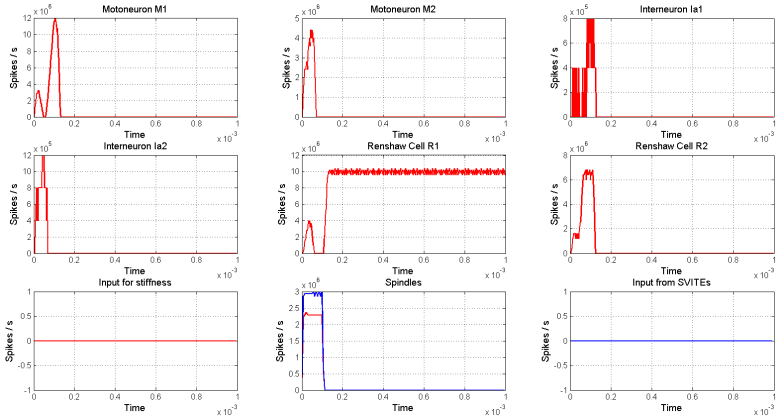

Figure 5.17: Reflex movement simulation. Immediately after the spindles show a short activation, the motoneurons of both muscles show a short activation, too. The activity of the motoneurons will be removed by the activity on the Renshaw and Interneuron cells. The motoneuron activity will provoke the reflex movement.

GO stimulus, as it was said in the original algorithm, the target is reached faster.

5.2 Hardware Results

5.2.1 Hardware platform: FPGA results

The model proposed was implemented using the AER Node board [106]. This board includes a Xilinx Spartan-6 LXT 1500 FPGA. It was developed by RTC lab under the VULCANO project[1] and it allows high speed serial AER communications over Rocket IO transceivers, and adaptation to particular scenarios through daughter boards connected on the top. For these tests, the setup includes a daughter board with an USB microcontroller that communicates with the FPGA over Serial Peripheral Interface (SPI). This interface is used to send the parameters needed for each block [41] and to

[1]VULCANO Project: Ultra-Fast Frame-less Vision by Events. Application to Automation and Anthropomorphic Cognitive Robotics. (TEC 2009-10639-C04-02)

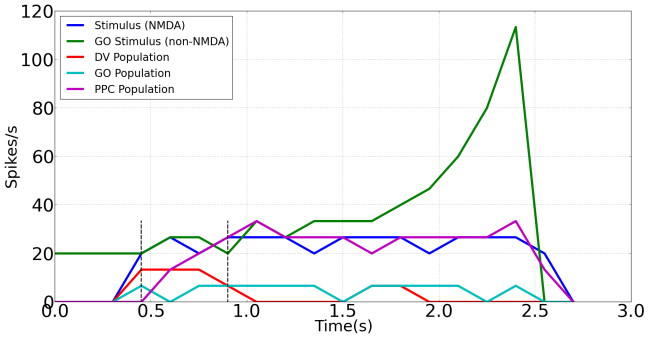

Figure 5.18: Firing rates of one neuron of each population. The target stimulus is 25 spikes/s. Rates of the stimuli, DV population, GO population and PPC population are shown according to the legend. The PPC population reaches the target stimulus set in 0.45 seconds since its activation as is shown by the vertical dotted lines. Once the target is reached, the DV population is fully inhibited and if the stimulus is not supplied, the network activity does not produce any spiking activity.

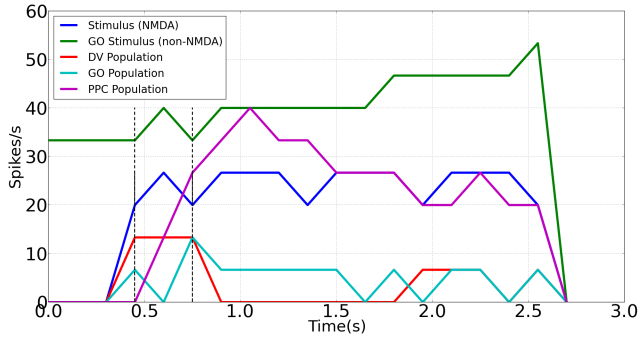

Figure 5.19: Firing rates of one neuron of each population. The target stimulus is the same at the previous plot, 25 spikes/s but the slope profile is higher than the previous one and now, the target is reached in 0.3 seconds since its activation. Once the target is reached, the DV population is fully inhibited and if the stimulus is not supplied, the network activity does not produce any spiking activity.

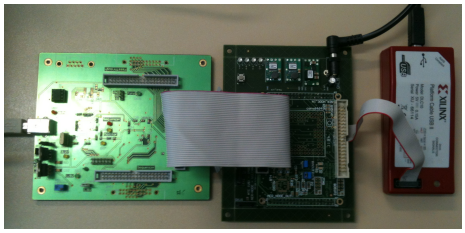

Figure 5.20: Hardware setup. It consists of a monitor board (left), AER Node Board (centre) and the programming tool (right).

manage the tests.

This main board also runs a massive spikes monitor [107] that addresses each block of the algorithm and does the handshake to communicate, using the AER protocol, with a monitor board [108]. The monitor board receives the spikes and allows them to be processed by the computer using jAER [109] or MATLAB. All the setup elements are shown in Figure 5.20.

GO Block behavior

Once the behaviour of the block is checked with the simulations results, figure 5.21 shows the tests done with the hardware platform. Table 5.2 summarize the data used to perform the tests.

Figure 5.21 remains showing the effect of the I&G included in the spike-based LPF. Figures 5.22 and 5.23 show the results considering slopes of 0.1 % and 10 % with a range of bits at the implementation of the I&G.

The results performed and showed with the wide range of bits (16, 20-24, 27-31) for the counter at the I&G, sometimes, show a jumping behaviour due to the saturation of the integrator. The higher the number of bits used, the slower the behaviour. There is a trade-off between the desired speed and the avoided saturations.

Table 5.3 shows the number of bits to implement the I&G with when the slope is changing its value.

Table 5.2: Data used for hardware tests.

Tests	**Slope_counter**	**Slope**	**Second per slope unit**	**Time to saturation (seconds)**
1	2,5 M	20	0.05	1
2	5 M	10	0.1	2
3	7,5 M	6.66	0.15	0.33
4	10 M	5	0.2	4
5	12,5 M	4	0.25	5
6	15 M	3.33	0.3	6
7	$2^{24} - 1$ M	2.98	0.33	6.71

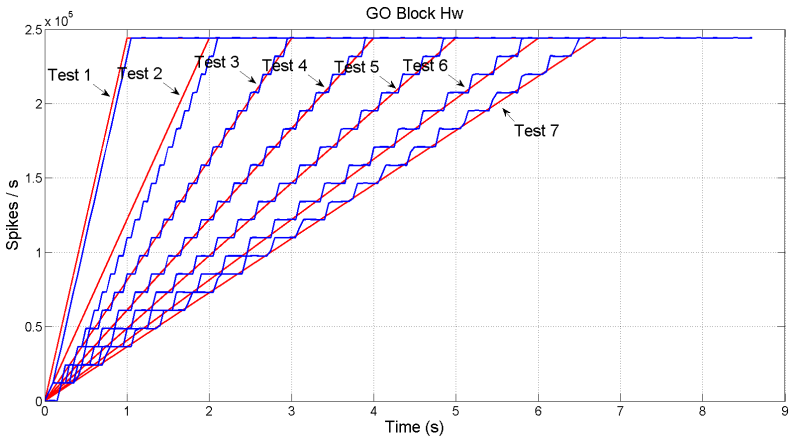

Figure 5.21: Results for hardware tests are shown. The red line represents the theoretical behaviour and the blue line represents the hardware results. The input for the generator was 8. It means an output frequency of 12,207 Kevents/s and multiplied by 20 in saturation it yields a total of 244,14 Kevents/s reached at different times at each test. The table below shows the parameters for the test. They are matched by the test numbers.

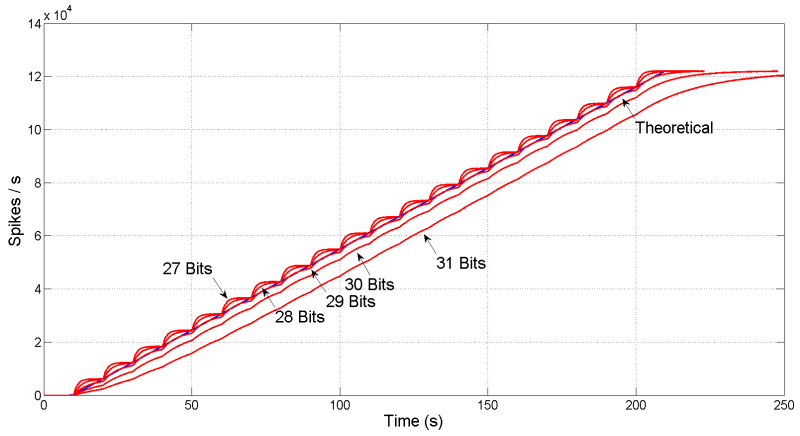

Figure 5.22: Comparison between five different bits implementations (27-31 bits) of the integrate and generate in the low pass filter (red lines). The theoretical behaviour is also represented in blue. Input is 6.1 Kevents/s. The slope is fix at 0,1 % and the saturation was fixed at 12.2 Kevents/s. Under these conditions, 29 bits is the best option.

SVITE deterministic

This subsection presents the results for the whole algorithm design (Figure 4.6 at previous chapter) adding a spike generator [110] as the target supplier (Figure 5.24)

The results shown in Figures 5.25, 5.26 and 5.27 show the evolution from the original VITE [72] to the translation version into spikes: SVITE.

Figures 5.25, 5.26 and 5.27 show the expected behavior of the algorithm translated into spikes paradigm against the behavior of the original design of the algorithm by Grossberg. The dotted lines are taken from simulations of the original VITE algorithm; solid lines show the same data but measured in the boards with the AER monitor [107]. The speed profile is taken before the integer block (the Integrate and Generate block in Figure 4.6) and the position orders at the output. The accuracy for both signals is highly precise and it suggests the opportunity of succeeding with a fully spike-based robot

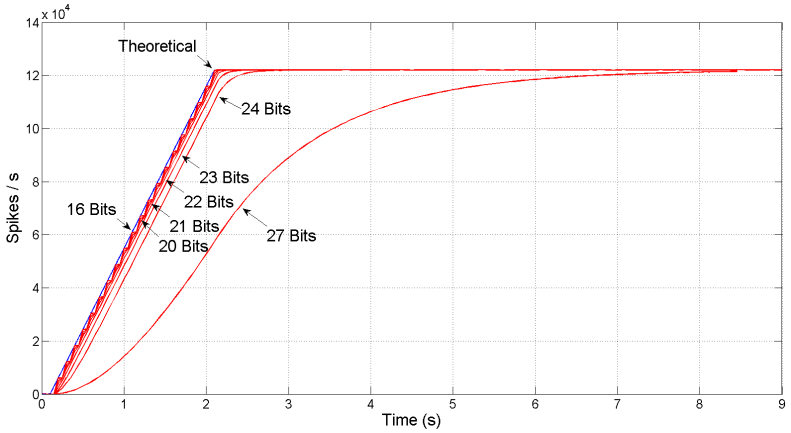

Figure 5.23: Comparison between five different bits implementations (27-31 bits) of the integrate and generate in the low pass filter (red lines). The theoretical behaviour is also represented in blue. Input is 6.1 Kevents/s. The slope is fix at 10 % and the saturation was fixed at 12.2 Kevents/s. Under these conditions, 21 bits is the best option.

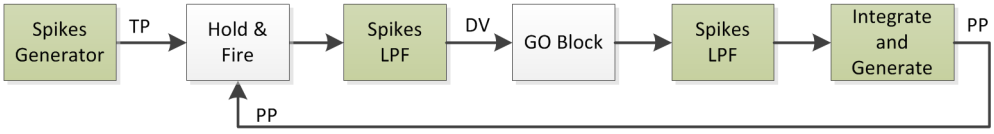

Figure 5.24: Setup simulation scenario to check the SVITE algorithm behaviour.

controller. Figure 5.28 shows simulation data for the speed profile achievable when the parameter slope_counter in GO block goes through different values causing 10, 50, 100, 500 and 1000 percentage slopes. The bell shape profiles confirm the studies in [99] where it is said that as faster is the movement the higher asymmetric speed profiles are performed.

Hardware Resources Consumption

Table 5.3: Trade-off between the number of bit to use at the implementation of the I&G and the slope that the GO Block is able to follow.

Implementation bits I&G	**Slope**	**Slope_counter**
9	50 M % - 390625 %	2^0–2^7
16	195312 % - 100%	2^8–2^{19}
21	10%	$2^{20} - 2^{22}$
26	1%	$2^{23} - 2^{26}$
29	0.1%	$2^{27} - 2^{31}$

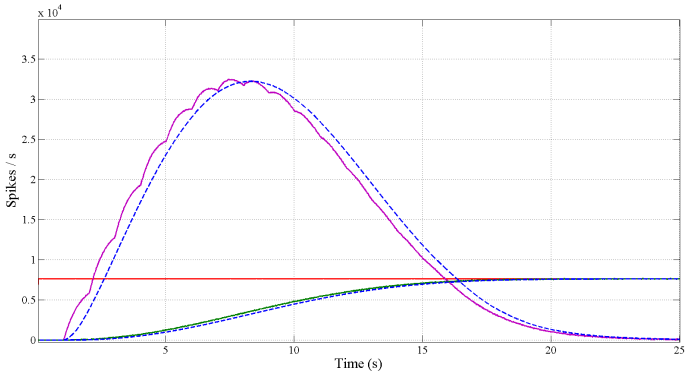

Figure 5.25: Performance achieved corresponding to one percentage slope in GO signal. Dotted lines are simulated in front of measurement solid lines. The bell shape profile signals represent the speed. The ripple in the spike-base behavior is due to the function that transforms the spikes into a continuous signal. The target is the same for both simulated and measurements signals and it is represented as a firing rate. It takes a total of 17 s to reach the target if we look through the position.

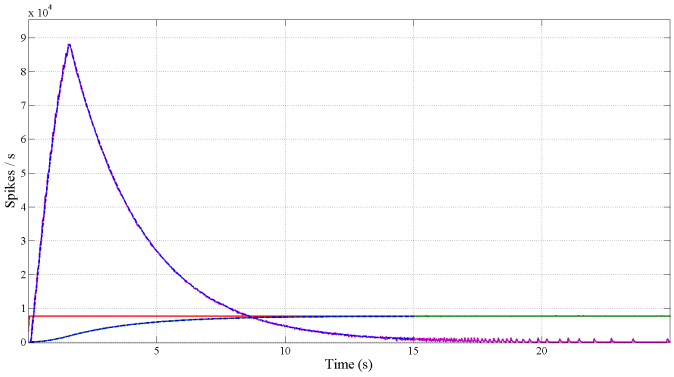

Figure 5.26: Performance achieved corresponding to ten percentage slope in GO signal. Dotted lines are simulated in front of measurement solid lines. The bell shape profile signals represent the speed. The ripple in the spike-base behavior is due to the function that transforms the spikes into a continuous signal. The target is the same for both simulated and measurements signals and it is represented as a firing rate. It takes a total of 11 s to reach the target if we look through the position.

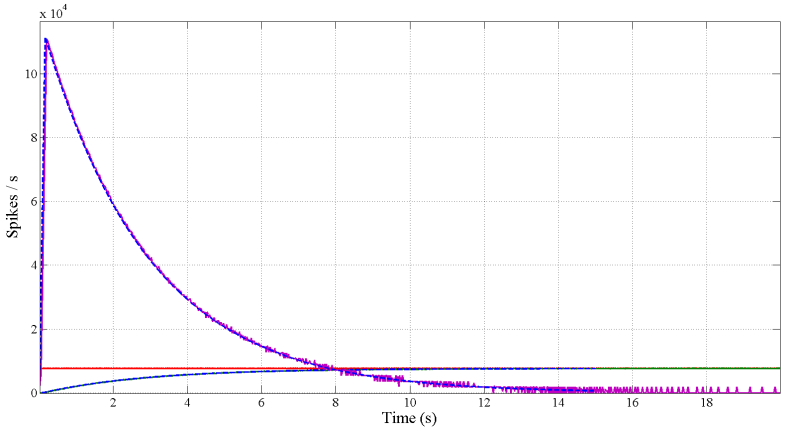

Figure 5.27: Performance achieved corresponding to 100 % GO signal slope. The bell shape profile signals represent the speed. With this high slope, the ripple in the spike-base behavior is more significant than in the others. It takes a total of 9 seconds to reach the target if we look through the position.

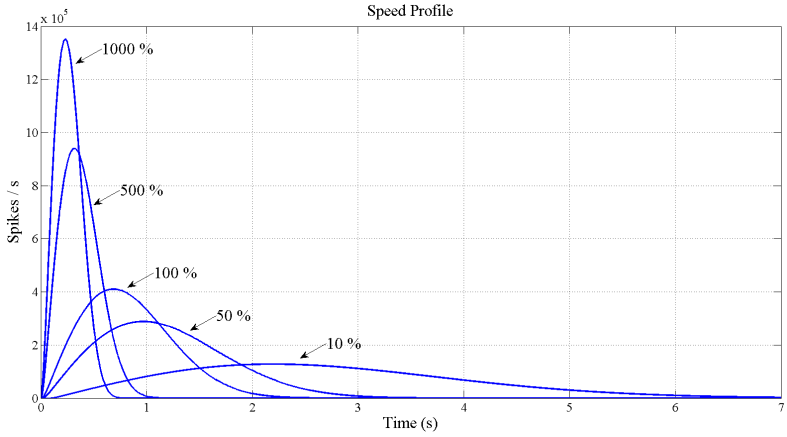

Figure 5.28: Speed profiles achieved by modifying slope_counter parameter of GO block. Making a comparative between slow and fast movements we can appreciate that the peak velocity is reached later for faster movements if entire length is considered.

Table 5.4: Hardware resources consumption by the Spartan 6 1500 device.

	Number of Slices	**Max. Blocks in the Device**	**Use by One Block**
Algorithm	238	96	1.033%
Algorithm plus monitor	533	43	2.31%
Algorithm plus interface	242	95	1.05%
Algorithm plus monitor and interface	537	42	2.33%

In general, to measure the hardware consumption in a FPGA, two points should be considered: the dedicated resources included to build up complex devices such as multipliers and the configurable logic blocks (CLBs) for general purpose. The algorithm does not use any complex structure. It just needs counters and simple arithmetic operation resources. Therefore the measurements are focused into the available slices at the FPGA.

We have synthesized the algorithm, including a spikes rate coded generator [110], the spikes monitor [107] and the interface with other neuromorphic chips for a correct debugging and a useful integration. Table 5.4 presents the data for the device with the report obtained.

In this table, the first column describes the element implemented for each case. The next column shows the amount of slices needed to synthesize the units at the FPGA. The following column represents the maximum number of units that could be allocated inside the FPGA. Finally, in the last column the total capacity of the device for all the synthesis performed is shown.

The results evidenced a low hardware resource usage when an isolated algorithm is implemented, just one per cent. Also, it is remarkable that the interface with other neuromorphic chips almost does not provoke an increment in the hardware resources consumption (only four slices). Consequently the final implementation for a complete architecture will consist of the algorithm and the interface. However, the design and test phases need the monitor in order to check the right behavior of the algorithm.

All the results presented in this section correspond just to slice consumption. The FIFO included within implementation uses dedicated memory blocks already present in the device, so it is not computed.

If we compare the maximum number of algorithms that can be allocated at the FPGA (that corresponds to the degrees of freedom (DoF) controllable in our architecture) with the iCub Robot necessity [111], it shows a great advantage using our approach. We can control up to 95 DoF (without monitor) in comparison with iCub platform which allows 53 DoF.

SVITE pseudorandom

The blocks under tests are the ones in green at Figure 5.24: the spikes generator and the I&G. From the border of the algorithm, a random component should be included to make sense of pseudo-random controller. So, to modify that behavior, we have included a linear feedback shift register (LFSR) first for the Spikes Generator block and later on for the I&G blocks. This component, with the proper XOR feedback, generates a uniform sequence of different pseudo-random values within a range that depends on the number of bits used to implement it. Once all the possible numbers have been generated, this LFSR will repeat the same random sequence over and over again. Also, a time-slice or bin to trigger the output can be configured. With this component, the probability to fire a spike at T will be defined by Equation 5.7.

$$P(spike = 1|T) = \frac{input}{(2^{(NBITS\_LFSR)} - 1)} \tag{5.7}$$

Some specific details depend on the type of block:

a. Spikes Generator: When the time bin is reached the value from the LFSR will be compared with the reference and only if it is lower, the block will fire a spike. Usually, the digital input reference for the spikes generation has 16 bits, so we took this initial number of bits to build the LFSR register. From Equation 5.7, one can see that there is an inversely proportional relationship between the reference (input) and the number of bits of the LFSR (NBITS_LFSR). A trade-off should be reached. Later on we will measure different behaviors for different LFSR lengths.

b. Integrate & Generate (I&G): In this case, the comparison with LFSR will be made with the number of spikes counted and only if it is lower, the block will fire a spike.

If we consider Equation 5.7 and the algorithm architecture, one can see that if we fix the same LFSR register for each I&G neuron included at the algorithm (one at each spikes low-pass-filter and last I&G), the firing rate will decrease along the blocks because the events (firing a spike) are not independent from the previous one, it is a conditional probability. Furthermore, we have to manage an adequate amount of spikes to allow the motors to run. To reach this behavior, we considered a gradual increase of the LFSR resolution which provokes higher rates of spikes for the useful signals.

At the end of the tests, we have a set of firing rates or spike signals belonging to each block of Figure 5.24 for all the combinations performed. First, they are split and each inter spike interval (ISI) is calculated. Second, for each ISI, we get the empirical cumulative distribution function (CDF) that can be compared with the ideal distribution's CDF.

On the other hand, the second comparison is carried out with the Kolmogorov-Smirnov test (KS test) and chi-square test to validate the distribution fitness. We have applied this test to compare how well the observed distribution of ISIs follows the theoretical exponential and gamma distributions. These tests give the p-value that allows rejecting or not the null hypothesis. We have considered a p-value between 0.05 and 0.1 as reasonable and above 0.1 to accept the hypothesis. Additionally, if the statistical test result is below 5% one can assume that the sample data fitness is good enough. All these statistical studies are conducted using the Statgraphics Centurion software package.

We performed these two comparisons because the KS test does not show a good performance at Null-hypothesis and p-value when the CDF for the theoretical distribution is made up from empirical data [112]. In such cases, the KS is an alternative statistic available in the test results. It shows the biggest difference between both distributions. Furthermore, with this value and the plotting comparison, a right conclusion can be drawn.

In view of this comparison, it cannot be forgotten that these signals are used for motor control purposes. Therefore, for each modification done to include the random element, we must check if the motor controller performance and accuracy are not lost.

The results achieved are also divided within the type of block:

Spikes Generator: If we were considering a bigger time bin, ie 0.1ms, instead of the agreed 20ns, we would only have an average of 67.4 spikes per second with the maximum digital input reference of 7000 and 20 bits LFSR to reach good distribution fitness for the "Spikes Generator". Therefore the time bin has been fixed to 20ns. Figure 5.29 shows the relation between the digital input references, the size (bits) used for the LFSR register and the goodness of fit for a gamma or exponential distribution for the "Spikes Generator", first block of the control model.

Figure 5.29 shows that for small sizes of the LFSR register (such as from 16 to 20 bits), the results of the test are far from the statistical threshold of 5%. If a deterministic source were used, the ks minimum values would be 0.6321 and 0.5243 for the Exponential and Gamma distributions respectively. As the size increases, the results improve. Besides, the higher the reference, the higher the spike rate, and so, the better the result. That matches the probability definition we made in Equation 5.7. So, in order to make a decision on which value should be selected, we are going to look at the p-value for each approach.

From the point of view of the Kolmogorov-Smirnoff test, the best approximation is got with a 30-bit LFSR, for a Gamma distribution. Looking through the statistical analysis, the p-value also indicates that the best fitness was obtained for the Gamma approach (Table 5.5) and for a 30-bit LFSR. One can think that an Exponential approach is more accurate, because it involves a Poisson process since there is a time independent probability definition for each event (spike in our case). However, the results show that the value produced by the LFSR register is not purely random, but pseudo-random. Therefore, both approaches are quite similar, as shown in Figure 5.29. The final decision is to take a 30-bit LFSR based on the p-value and KS statistical value. Figure 5.30 shows also the accuracy for a 24-bit LFSR.

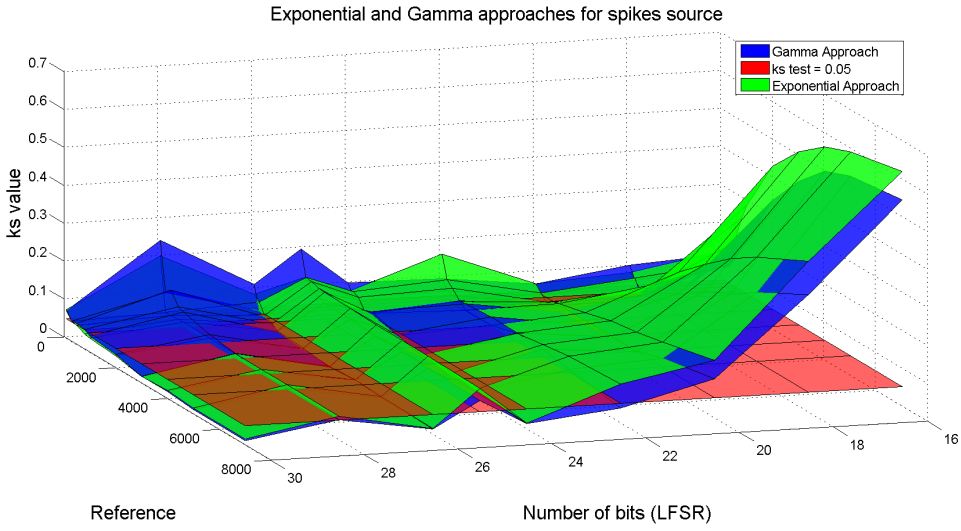

Figure 5.29: 3D representation of the ks_value for the gamma and exponential distributions. The red plane is fixed at 0.05 (threshold of the K-S test: passed if the value is below the threshold). The green surface represents the Exponential approach and the blue surface represents the Gamma one. It can be observed that for LFSR resolutions of 24, 26 and 30 bits the test is passed for both approaches (the red plane is visible).

Table 5.5: P-value and average firing rate for the exponential and gamma approaches when the LFSR is implemented with 24, 26 and 30 bits.

	p-value (exp)			p-value (gamma)			Average(exp) [spikes/s]			Average(gamma) [spikes/s]		
Ref.	24 Bits	26 Bits	30 Bits	24 Bits	26 Bits	30 Bits	24 Bits	26 Bits	30 Bits	24 Bits	26 Bits	30 Bits
100	0	0	0.219638	0	0	0.258212	50.57	37.25	2.33	130.47	32.2	3.03
300	0	0	0.316934	0	4.69E-08	0.314826	447.27	111.68	6.94	387.76	125.84	6.62
500	0	0	0.0539985	0	0	0.0239918	745.17	186.29	11.59	718.45	191.92	10.86
800	0	0	0.228294	0	0	0.0519524	1191.95	297.94	18.43	1201.62	300.08	17.15
1000	0	1.23361E-05	0.0189995	0	7.79468E-05	0.00416151	1487.61	372.81	23.32	1527.63	363.5	22.63
2000	0	5.30E-07	0.00278128	0	1.24587E-05	0.000501784	2978.14	745.2	46.55	3119.73	762.55	44.72
3000	0	0.0108252	0.223811	1.6556E-06	0.00204732	0.189354	4462.33	1118.03	69.86	4802.4	1102.85	69.42
4000	1.2414E-06	6.0598E-06	0.0162689	0.00267274	3.18256E-06	0.0587112	5940.60	1489.4	92.95	6422.73	1481.73	89.04
5000	1.7624E-05	0.000336263	0.441025	0.0012794	0.015823	0.8107	7435.44	1861.94	116.29	8013.46	1905.94	113.88
7000	0.00764807	0.00480427	0.134655	0.0197746	0.0138585	0.535872	10409.17	2605.19	163.02	10959.4	2668.41	159.20

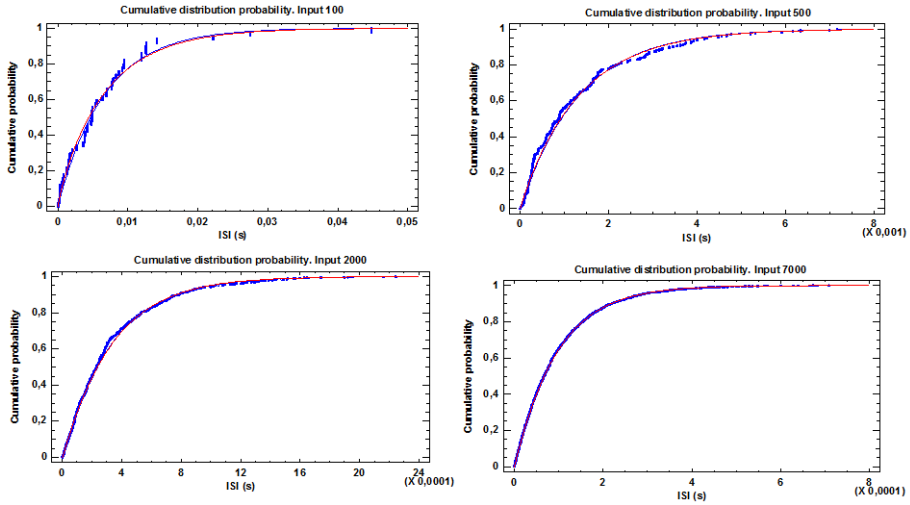

Figure 5.30: Cumulative distributions function for gamma and exponential approaches when a 24 bit LFSR and constant digital input reference values of 100, 500, 2000 and 7000.

If we go across the Table 5.5 and compute the average value, it results in 0.16 for Exponential and 0.22 for Gamma, for 30 bits value. For 24 and 26 bits, the average p-value does not reach 0.01. Looking at columns where the average firing rate is shown, one can see that for 24 bits there are bigger differences between the gamma and exponential approaches and if LFSR size is higher, the differences are lower, reaching at 30 bits value nearly the same rates.

Considering a 30-bit LFSR register, there is a linear relationship between the reference delivered to the block and the output firing rate for the Gamma approach. The maximum firing rate within a 16 bit width reference (65535) is 1.5 K spikes per second.

With the "Spikes Generator" characterized, it is time to use it inside the neuro-motor controller, study the behavior and compare it with the previous deterministic one. Figure 5.31 shows the output signals for each block of the algorithm (blue lines). It can be observed that if we are using a random source, the commanded signal does follow the input signal multiplied by 2. That is the reason why there are three different representations in the graph. Eventually, it is possible to use a random "Spikes Generator" to provide the reference to the algorithm.

Integrate and Generate (I&G): We have started using the same LFSR for each I&G neuron. Figure 5.32 shows the ks-values for the useful blocks from the motor control point of view, that is the speed profile signal and the commanded position signal. It is demonstrated that adding the same LFSR to all the neurons is not a good option; Figure 5.33 shows how the algorithm goes into a non-stable situation and starts oscillating around the goal; eventually, the goal is not reached. The reason for this is that with such a small number of spikes flowing across the algorithm, the inhibited connections (understood as spikes which decrease the integrated value) can have higher and so damaging effects on the whole algorithm. The response of some neurons may start oscillating due to this negative contribution. Also, it seems that if the GO block injects a large number of spikes, those damaging effects will be amplified.

To increase the firing rates as it was defined in the methods, some of the tested options have been: 10-15-20, 15-20-25 or 20-25-30 bit LFSR (each

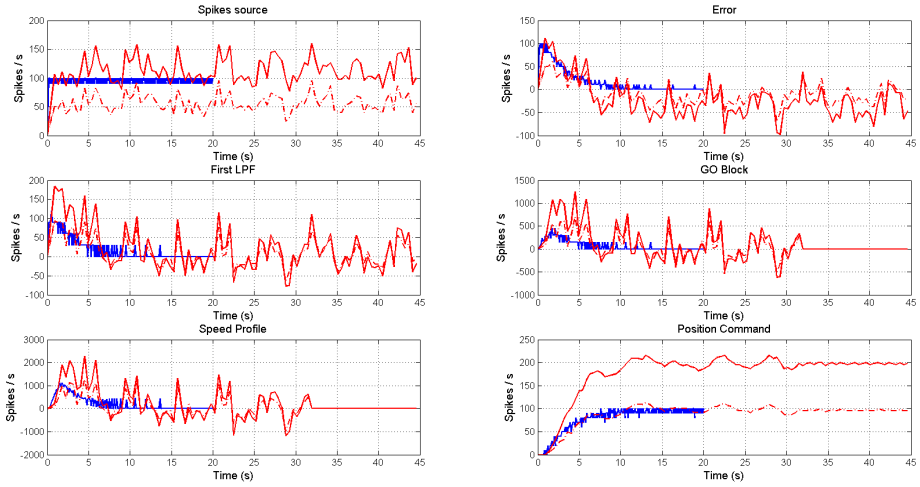

Figure 5.31: Output signals of each block of the algorithm. They were obtained by integrating a fixed period of spikes in a similar way as the kernel density estimation [7]. Blue lines represent the behavior used in the running neuro-motor-controller presented in [8] versus the random source, represented by red lines. The dotted red lines represent a 50 spikes/s input reference and the solid red lines represent a 100 spikes/s input. The standard deviation calculated from comparing the sources was 7.4 spikes/s; 100 spikes/s for the speed profile and 3.86 spikes/s for the commanded position.

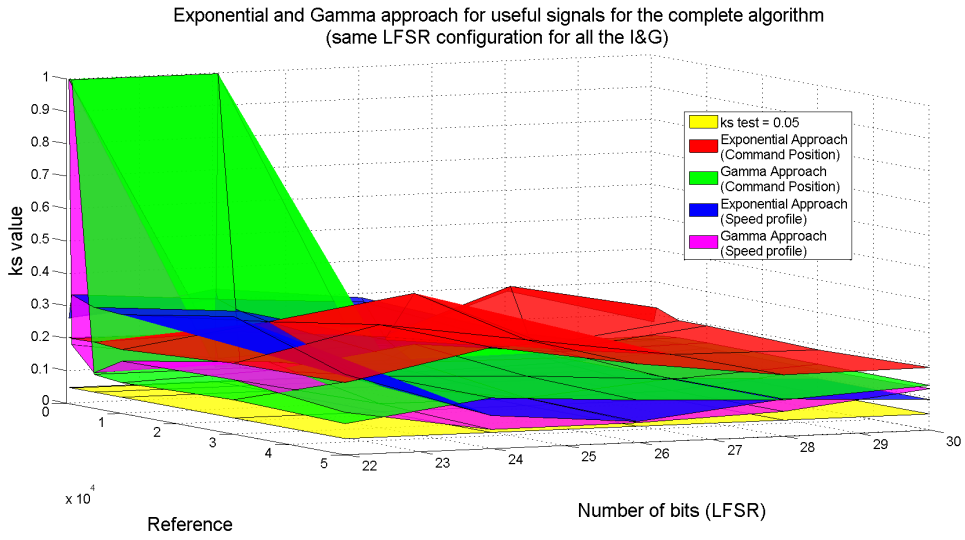

Figure 5.32: The yellow surface represents the plane for ks = 0.05. The green surface represents the ks-value for the gamma approach and the red surface represents the ks-value for the exponential approach; both for the commanded position signal. The blue surface represents the ks-value for the exponential approach and the pink surface represents the ks-value for the gamma approach; these two for the speed profile signal. As it is easy to see, none of the combinations passed the test, neither any of the possibilities between the range (16, 32) bits for the LFSR got a p-value different from zero.

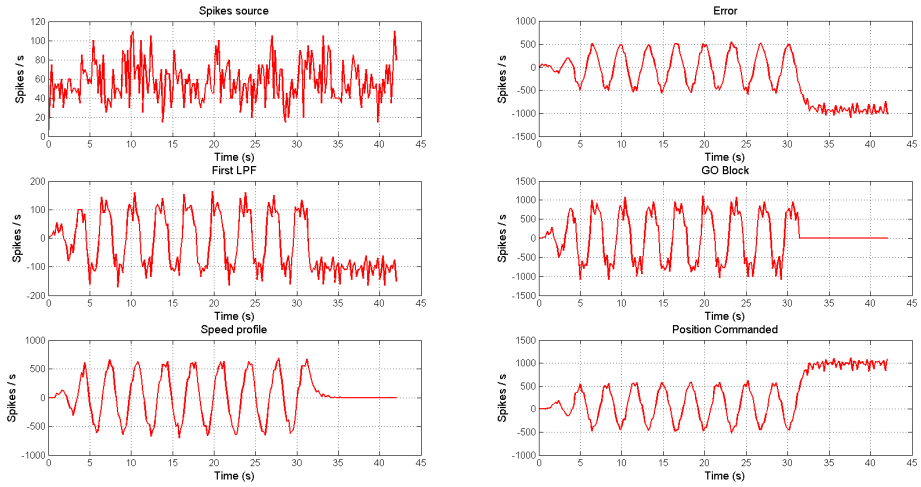

Figure 5.33: Output signals of each block of the algorithm. These signals are for a 25 bit LFSR register for all the neurons and 50 spikes per second as the input reference to the algorithm.

LFSR of the triplets for each I&G component: both spike low-pass-filters and last I&G block); other configurations do not provoke any spikes at the useful signals (speed profile or position). The obtained results open up discussions:

The first two configurations do not pass the ks test for any digital reference or any useful signal. Therefore, although the signals nearly match the deterministic behavior, we cannot predict them with any distribution.

However, the last configuration (20-25-30) gives us a chance because if we compare the signals behavior with the complete deterministic situation, the result is an absolute equality (Figure 5.34). Thus, these signals (speed profile and command position) are useful for motor control. But, can we predict them? Do they follow any well-known distribution?

The statistical study states that position commanded signals can be approximated by an exponential distribution for references higher than 1,600 (since 16 bits are used for the input, the range is very large).

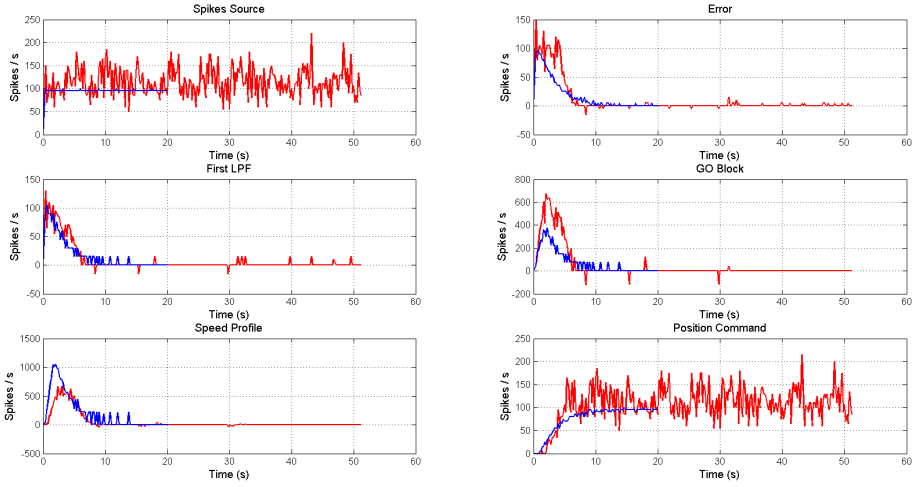

Figure 5.34: Output signals of each block of the algorithm. The blue lines represent the entire deterministic algorithm and the red lines represent the one which includes the random "Spikes Generator" and the 20-25-30 LFSR configuration. The average behavior is nearly the same. The standard deviation calculated from comparing the sources was 21.38 spikes/s; 157 spikes/s for the speed profile and 11.53 spikes/s for the commanded position.

5.2.2 Hardware platform: FPGA + Robotic Platform results

The robotic platform used is a stereo-vision robot with four degrees of freedom powered by DC motors. Although, the motors have an isolated movement, at this moment, they are coupled in pairs, one for each axis. Thus, each axis is fed with one algorithm, so we have one algorithm for the pair of motors of the axis. The power supply requirement of the motors is 24 Vdc. The manufacturer of the motors is Harmonic Drive and the model is RH-8D6006. The structure of the robotic platform is made so that the motors of the y-axis are crossed to their axis and have a transmission belt to move the arm. With regard to this structure, we fed the y-axis motors with the position (because it is needed to hold the spike transmission) and the x-axis motors with the speed profile.

We propose to use Pulse Frequency Modulation (PFM) to run the motors because it is intrinsically a spike-based solution almost identical to the solution that animals and humans use in their nervous systems for controlling the muscles (See Appendix A). Nevertheless, we need to adapt the spikes because the digital clock of the boards is fixed at 50 MHz resulting in a spike width of 20 ns and this signal is very fast and the spikes too short for the motors model of the robotic platform [41].

To compute the maximum and minimum spiking rate allowed we must look in detail at the board components and the DC motor, respectively. On the one hand, for maximum firing rate, the power driver of the motors consists of an optical isolator and H-bridges. Both components have some features regarding the switching frequency: for the H-bridge it is fixed at 40 KHz, but it is recommended to work at 25 KHz (minimum period of 40 μs) to avoid malfunctioning. As for the isolators, there is no maximum switching frequency defined, but two important temporal restrictions must be considered: 6 μs and 5 μs for raise and fall, respectively.

Merging these data, it results into a maximum firing rate of 25 KHz (40 μs of minimum period), and within this max rate the spike width can be solved. Using the minimum period of 40 μs and taking into account the temporal restrictions of the isolators, it results in a time period of 29 μs as maximum width. We have chosen a secure width of 25 μs for margin and to spread out

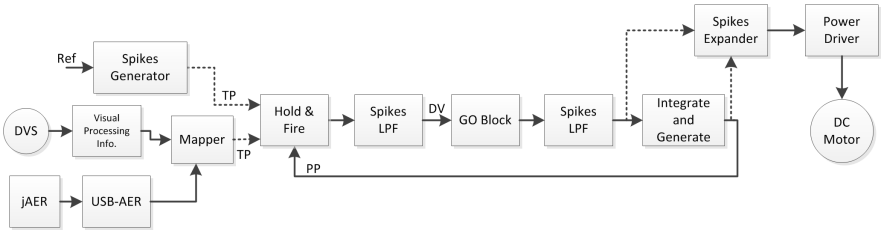

Figure 5.35: Block diagram of the setup to perform the tests.

the spikes up to 750 clock cycles. Definitely, with these data, the maximum switching frequency will be 25 KHz and the spike width 750 clock cycles.

On the other hand, to compute the minimum spiking rate allowed it is necessary to analyze the target actuator (DC motor in our case). DC motor acts as a low pass filter and the transfer function can be calculated using the parameters from the manufacturer [101]. This function and, particularly, its step response allow us to select the motor's minimum switching frequency (maximum period) suitable to follow an input properly. The step response calculated illustrates an approximate total time of 40 ms to follow the input. Therefore, we are going to select a lower order value with a little margin: 1 ms of maximum period, so a minimum frequency of 1 KHz for the incoming spikes.

These two limits will allow us to build up the empirical table that maps the reference supplied to the system and the movement produced at the platform. To sum up, we have the operating margin for the motors: from 1 KHz to 25 KHz and the spike width as 750 clock cycles. Notice that, if we make the spike injection in the GO block lower than ten percent of slope, it would not cause any movement at all because the motor will filter the spikes. In contrast, a much higher slope could saturate the system without a closed loop control.

Results

The setup prepared to run the tests is the one shown in Figure 5.36.

The target can be supplied by the PC using the jAER (synthetic sequence), using a DVS sensor or manually, using a Spikes Generator (Figure 5.35).

If the DVS is used, then a visual processing information layer is needed [2]; this stage will provide the center of mass of the target in the form of a (x,y) pixel via AER to the FPGA board where the algorithm is. This information will be matched with a firing rate to reach the desired position by the mapper block and eventually, the final rate will be delivered to the algorithm.

When the DVS is in used, we hace a precise match with the biological behaviour described previously. The DVS plus the visual information processing layer will play the role of the somatosensory areas where the stimulus is generated. Then, the mapper performs the same task of the PPC: translating the visual information into the framework understood by the actuation layer.
Then, the first part of the algorithm consisting of the H&F and the LPF will play the role of the M1. There follows the GO Block who plays the role of the BG. Finally, at the end of the chain, the I&G plus the spikes expander could be seen as a spinal cord where the motoneurons are.

If a synthetic sequence is supplied, a special board [108] is needed. This board will transform the information from the jAER at the PC to an AER flow that will be supplied to the FPGA.

The results presented in this subsection used a DVS sensor to supply the targets. Thus, a full neuroinspired controller, where all the information flowing are spikes, is achieved (Figure 5.36 shows the setup).

The algorithm will be replicated as many times as the DoF of the robot. In Figure 5.35 only one DC motor is represented to avoid cluttering.

One of the more attractive items of the algorithm is that it is possible to generate synchronous movements by controlling the GO signal independently for each motor. Figure 5.37 shows the real measurements of the position reached when the target is fixed at (125, 90) in the frame of reference of the retina. This becomes an angle of 75 degrees for the x-axis and 48 degrees for the y-axis in the frame of reference of the robotic platform. Moreover, the figure shows the translation of the target delivered by the sensor.

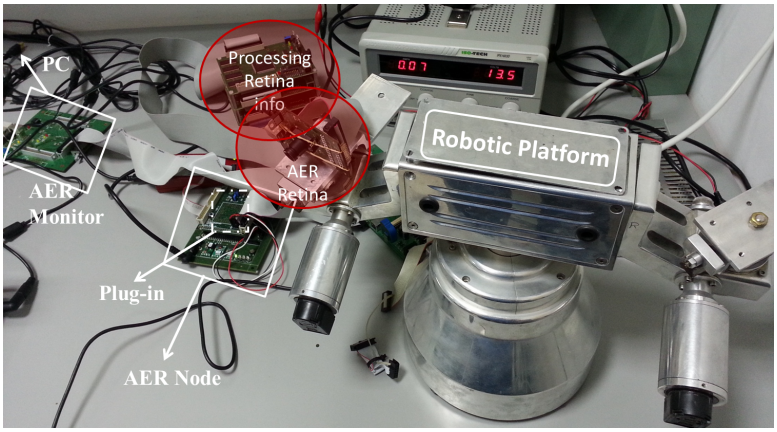

Figure 5.36: Robotic hardware setup. It includes all the elements to check the behaviour of the algorithm proposed.

Since our robotic platform has a special architecture that requires us to use the position commands for the y-axis and the speed commands for the x-axis (to hold the target position at the end) it is not easy to produce synchronous movements; indeed it is impossible because the position commands are slower than the speed commands. Nevertheless, we have fed the y-axis also with the speed profile (although it does not hold the position at the end) to check the synchronicity of a movement done with this algorithm; the result is extremely accurate if we compare it to the target provide representation.

Turning to the slower reaching, the x-axis reachs the position commanded in approximately one second and then starts the movement in the y-axis. If we compare the theoretical signal delivered to the motors and the movement achieved (reads out from the encoders of the motors and using jAER tool [109]) shown in Figure 5.37, a couple of comments regarding the reaching time appear. For the x-axis, the one commanded by the speed profile, it is quite different, and for the y-axis, commanded by the position, we found nearly the same reaching time. The reason for the difference at this reaching time in the x-axis can be understood with these two points: non-feedback used, which means once the motor starts running we do not have any inertia control and non-feedback from any sensor.

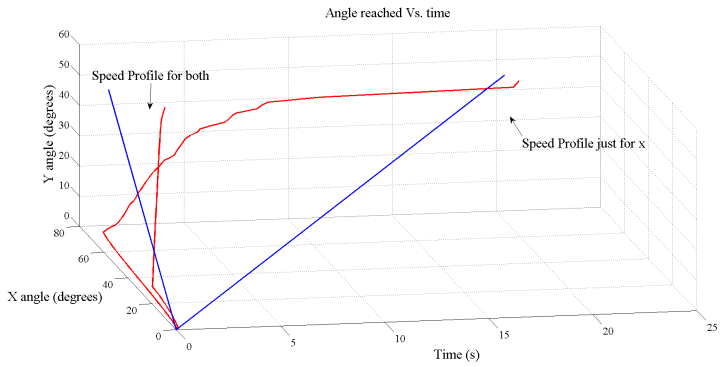

Figure 5.37: Angle vs. time reached for both axis with (125, 90) input. The retina has 128 x 128 pixels. The red lines show the trajectory followed by the robot when we used the speed profile for both axis and just for the x-axis; the blue lines represent the motion delivered by the DVS sensor.

However, the accurate movement achieved when both axis use the speed profile requires us to think about obtaining gestures if all the motors were able to be fed with the suitable velocity profile to hold the position.

Thus, in Figure 5.38 we have performed a test to check the tracking properties of the robot. The test is done just for the x-axis which is the one commanded by the speed profile. For these tests, the targets have been delivered to the FPGA board by DVS sensor with three difference time distances in between: 2.6, 3.9 and 5.2 s for each test. It is possible to detect the latency of 0.1 s at the beginning. The processing time can also be calculated by computing the time between the target delivery and the start of the motion, minus the fixed latency. It results approximately 0.5 s.

The difference between the angle reached by the robot and the motion represented is due to the resolution obtained by the encoders of the motors and also to the cumulative error in an open-loop controller within many targets without calibration between them. Finally, empirical tests reveal an accurate tracking by the robot when the distance between target deliveries is at least 2 s. This data between targets depends on the actuator selected. If the DC motors have a fast response, this time can be reduced considerably.

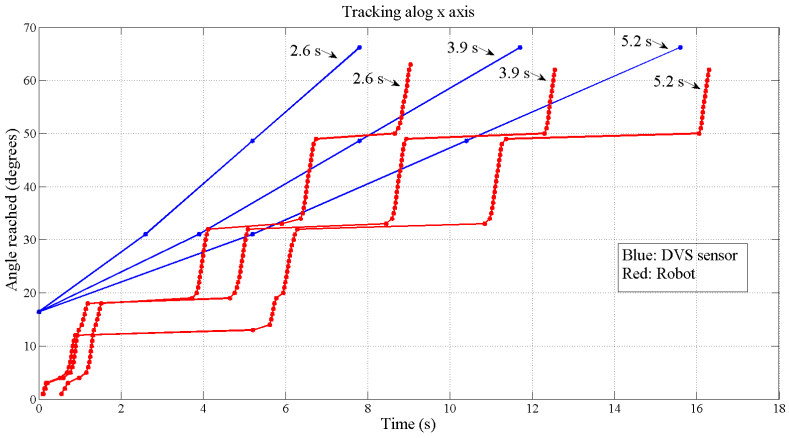

Figure 5.38: Angle vs. time tracked for the x-axis. The input is a go along the x-axis in the frame of reference of the retina. The red points show the angle trajectory followed by the robot and the blue points show the targets delivered to the robot

Feedback results

This section presents the results achieved with the implementation of the whole model of Figure 4.8 without the SFLETE interface element. In this case, the tests are done considering a small robotic arm: reference OWI-535 from the OWI brand.

Figure 5.39 shows the firing rates including the input and output signals to the feedback block. This block will make a comparison between the signal coming from the motor encoders and the input supplied from the SVITE algorithm. The result of this comparison will run the motors of the robotic platform. Since this robotic arm has three DoF, the SVITE algorithm is replicated three times. However, the plot represents the results of one of the algorithms.

Power Consumption

The power consumption of the design implemented can be divided in three different parts: the device static, design static and design dynamic power

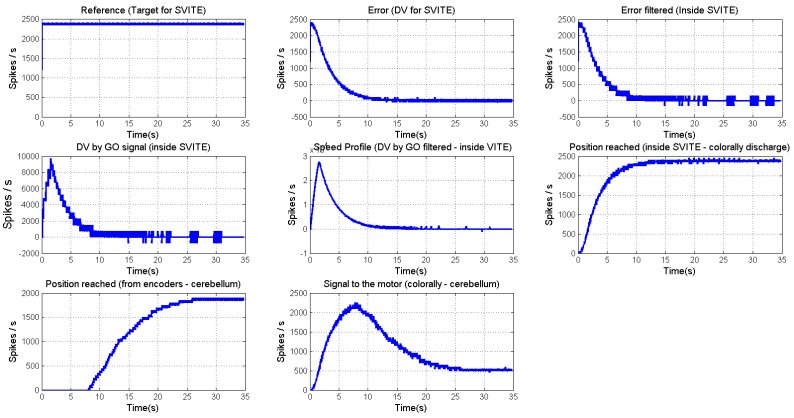

Figure 5.39: Output firing rates of each block of the algorithm plus the feedback. Last two ones represent the input from the encoders and the output provided to the robotic platform.

consumption. The device static power consumption is also called the off-chip power and it is referred to the power consumption of the board without any configuration. The design static power consumption is the power used when the design is just programmed into the board but it is not running. Finally, the dynamic power consumption is referred to the power used by the design when it is running.

We have used the XPower estimator tool from Xilinx to get the device static and design dynamic power consumption. The results are: 0.113 W for the device static power and 0.027 W for the design static power. The design dynamic power is obtained by computing the difference between the real measurement, when the algorithm is running, and the addition of device static and design static power consumption. The power consumption measured is 3.4 W, thus the design dynamic power is 3.26 W.

5.2.3 Hardware platform: VLSI results

The multi-chip setup used in this section is composed of two chips: one comprising 2048 neurons and another one comprising 128 neurons. The total number of analog AER synapses is 16384. Both the neural and synaptic

circuits exhibit biologically plausible adaptation mechanisms (e.g. short-term depression, spike frequency adaptation, etc.). Please refer to [19] for detailed descriptions of the circuits implementing the neuron and synaptic models. The neuron used in both chips, described in [32], is a compact low-power leaky integrate-and-fire circuit that implements spike-frequency adaptation as well as a tunable refractory period, and voltage threshold modulation. The neurons of these chips are tied to differential pair integrator synapses [19] that can be stimulated by means of external input pulses in the form of address events provided by a hardware infrastructure [113, 114]. Both chips have been fabricated using a standard 0,35 μm CMOS process. Figure 5.40 shows the multi-chips setup where the tests are done [2]. Finally, the neurosetup is connected to the AER Node board [106] using AER. Eventually, at the end of the architecture, the robotic platform can be allocated.

SVITE results

One chip is used for each population: DV, GO and PPC. This division has been done on behalf of a better parameter tuning; otherwise, it will be not possible to achieve an accurate performance due to shared parameter among different populations. The number of neurons will depend on the robotic platform to control because when PFM is used, the firing rates to drive the motors will be fixed by the motor model. Thus, the number of neurons is fixed by the required firing rate. With this technique, as we have already mention, the spikes can be supplied to the motors by spreading them the appropriate time length to avoid them to be filter by the motor and also a jerk movement (a jerk movement is a non-smooth one as the ancient 'robot' motion).

The tests are done considering the same small robotic arm (rerence OWI-535 and the brand is OWI) [44]. Specifically, for this robotic model, the number of neurons per population needed to achieve a smooth control of the robotic arm was 60 IF neurons. The fashion connection is one-to-one. It results in a 60 replicated layers of the network showed in Figure 4.7 without any connection between them (that is the reason why only one neuron per population was used for the simulations). Figure 5.41 shows the response

[2]This setup belongs to the Neuromorphic Behaving Systems Group led by Junprof. Dr. Elisabetta Chicca. Bielefeld University.

Figure 5.40: Multi chip setup used to perform the tests. The one marked with the '0' stick is a 1-D chip which comprises 128 neurons. The chips marked as '1' and '2' are 2-D chips which include 2048 neurons each. The daughters board connected to each chip are responsible to provide the communications.

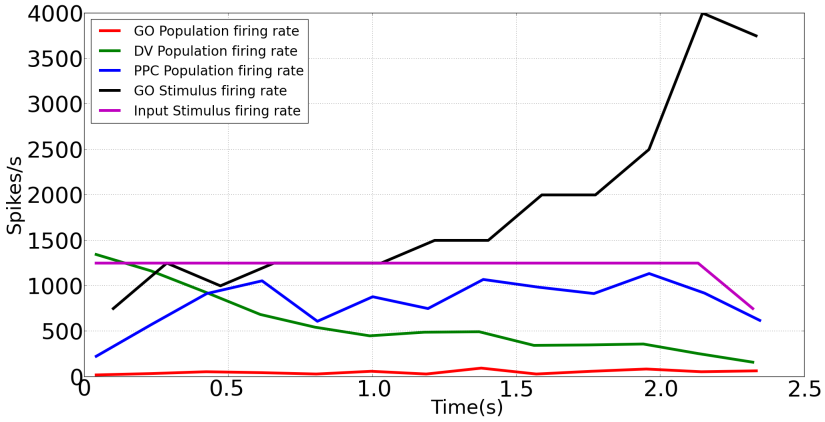

Figure 5.41: Firing rates of one neuron of each population. The target stimulus is 25 spikes/s. Rates of the stimuli, DV population, GO population and PPC population are shown according to the legend. The PPC population reaches the target stimulus set in 2 seconds. Once the target is reached, the DV population is fully inhibited and if the stimulus is not supplied, the network activity does not register any spike flowing.

of the hardware neural network which exhibits the same behavior observed in Figure 5.18 (simulated network). Also, in this case, no spike is fired before the input stimulus is present. The implementation of the co-activation NMDA and non-NMDA is done using an excitatory connection with short time depression and the synapse implemented on the chip which includes the NMDA behavior. If the slope profile of the go stimulus is changed, Figure 5.42 shows how the target is reached faster than the previous one keeping the input target with the same value and the same configuration for the population. The output activity of the PPC population is delivered to the FPGA to spread the spikes and eventually, drive the motor.

The main difference with the previous model is that the response at the output of the GO multiplication is not a bell-shaped speed profile as it was stated in [99]. However, the functionality of the GO signal was not changed: the higher its slope, the faster the reaching movement.

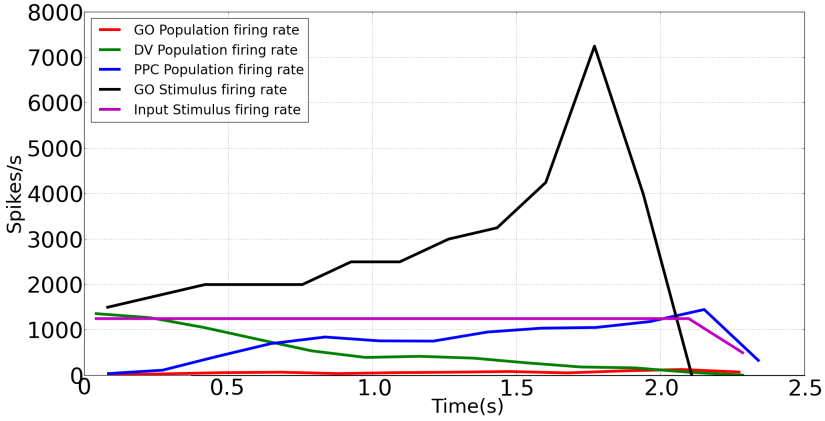

Figure 5.42: Firing rates of one neuron of each population. The target stimulus is the same at the previous plot, 25 spikes/s but the slope profile is higher than the previous one and now, the target is reached in 1.8 seconds. Once the target is reached, the DV population is fully inhibited and if the stimulus is not supplied, the network activity does not register any spike flowing.

If we would try to recover the bell-shape profile for the output rate of GO population the dynamic response of the network will be driven by the GO stimulus and also the co-activation of the movement within the NMDA channels will be lost.

5.3 Discussion

The spike-based method of motor control used in this dissertation, appears to be the most neuro-inspired one, because it enables delivery of the spikes directly to the motors. Neither computation nor translation is necessary to produce motion.

If we make a comparison between the two hardware platforms used, it can be seen that the spike-based algorithm implemented on the FPGA copies precisely the original algorithm from Bullock-Grossberg in front of the VLSI Setup implementation, which is a better option if the neuroprosthetic route is taken. On the other hand, the FPGA implementation is more robust and appropriate for industrial robotics if the final actuators are well measured to achieve a fast response.

If we look at the functionality of the GO signal: the FPGA implementation maintains its function as it was designed in the primitive algorithm. Thus, the SVITE algorithm can support the target delay [115]: if the GO signal is shot after or before the target is submitted, it will only cause a delay or a jerk due to the higher starting speed, respectively.

Conversely, the VLSI implementation has a stronger link with biology than the FPGA one, because of the co-activation of the NMDA and non-NMDA channels. The GO function also maintains its functionality. However, the bell-shape speed profile is not reached anymore.

The idea of using PFM in a neuromorphic environment to drive the motors gives us a huge advantage: it minimizes the effect of noisy or non-desired spikes. If the time length of the spikes is fine-tuned, isolated noisy spikes can be filtered by the motor.

Regarding the VLSI implementation, up to this moment, the main disadvantage of this model is the open loop controlling technique. However, even with this technique, using the output rate of the PPC population could be very interesting, as the population has a temporal rate.

If we look at the results presented in feedback section (Section 5.2.2) it can be seen a difference between the target and the position reached. This can be explained as a timing mismatch when the comparison is made by the feedback block or due to the encoder resolution. Also, the latency at the beginning of the motor response is due to the slow motor response. Thus, the accuracy of the close-loop algorithm depends on the robotic platform in use.
Furthermore, if we go back to the model shown in Figure 4.8, an improvement appears: the feedback block is appart from the planning stage where the algorithm is so, a new target could be supplied to the SVITE algorithm and it has not a precise information about the end-effector position. Has it reached the first target? To solve this question, a connection from the feedback block to the SVITE algorithm appears to update the present position.

The SFLETE algorithm gives us the opportunity to test the architecture with real muscles or robotic platform made of muscles. The point that remains uncertain is the relation between the activity of the motoneuron pool and the force produced. Usually, robotic platforms do not include elements in which forces are needed, but current consumption. The force of a DC motor could be matched with the current to drive it. The higher the current, the higher the force generated. But if we apply a higher current to a DC motor it will turn faster and a movement mismatch will occur. A load must be placed at the end effector to see the effect introduced by a higher current consumption.

Chapter 6

Summary, conclusions and future work

6.1 Conclusions

The conclusions achieved are the following:

- A full neuro-inspired motor controller using neuromorphic hardware is presented.
- The main bio-inspired algorithms were studied and the VITE and FLETE were selected as the most suitable to implement them.
- A model to control the movements and the forces is adapted to use under the spike paradigm of FPGA or VLSI chips.
- A new block is designed, simulated and implemented: GO Block.
- The full controller proposed is simulated: Both movement and forces controller under the FPGA spike-based constraints and the movement one under the VLSI constraints.
- A close-loop neuro-inspired motor controller is implemented on the FPGA. An end-effector (robotic platform) is added at the end of the architecture. The VHDL needed to enable an accurate functioning are generated.

- We have demonstrated that a Poisson "Spikes Generator" can be used to convert the digital input reference into spikes for a neuro-inspired neuro-motor-controller without losing accuracy.

- The complete neuromorphic system presented includes a full chain: from dynamic vision sensor, through spike-based cascade architecture of vision processing for object detection and tracking, to a neuro-inspired motor control algorithm implemented in the spike-domain (SVITE).

- An open-loop neuro-inspired controller is implemented on a multi-chip VLSI setup. The first attempt to use analog low-power subthresold VLSI IF neurons for motor controlling (without using a microcontroller) is presented.

- The architecture proposed is able to reach a supplied target with a range of stiffness without a big disturbance at the position reached.

- A method to select the parameters when a PFM modulation is used to run DC motors is presented. We claim that by using PFM to run the motors in a neuromorphic hardware environment, the noisy spikes will not affect the functioning of the system.

6.2 Future work

The present dissertation leaves open some exciting possibilities for future research such as:

- Study how to include a precise model of the cerebellum to the system. This would allow including learning techniques in order to tune the projections comming from higher layers.

- Design new features to add to the algorithms in light of recent neuroscience studies.

- Recent advances on robotic platforms led us to check the architecture designed on structures in which some elements mimic the human muscles. This would be the first step to take, then if the architecture succeeds, a move to neuroprosthetic field could be done.

- Design, test and manufacture of VLSI chips where neuronal circuits that perform specific motor controlling tasks are allocated.
- Include learning techniques to all the levels of the controller.
- Study how movement patterns can be stored and reproduced at any time, just with a stimulus trigger.

Appendix A

PWM Vs. PFM

This appendix comprises of a report about Pulse Width Modulation (PWM) and Pulse Frequency Modulation (PFM) modulation used to drive motors using spiking neural networks.

A.1 Introduction

Both PFM [116] and PWM [117] are two different processes to carry information through a communication channel. In general terms, a modulation includes a carrier signal modified according to the message signal where the information is allocated. The new modulated signal is ready to be transmitted.

Specifically, these two modulations are known as the methods to drive actuators like DC motors, servo motors or stepper motors. However, PWM is almost present in all of the situations where a motor is driven. It is large known and the most used [118–120]. PFM is still unknown; it is not so common to find a microcontroller or at least one small driver which implements a PFM modulation.

Also, if we look through the patents with the tool from Google [121], the difference between the popular PWM and the PFM can be seen: 2.01 million for PWM and 1.28 for PFM.

A.2 PWM

A.2.1 Introduction

Pulse width modulation uses a fixed frequency squared waveform to carry the data. The information to carry is an analog value. The modulator will spread each pulse according to the analog value. Thus, the width of each pulse will have a relationship with the analog value to transmit. The period of time of the signal with a high level it is called the duty cycle. Therefore, duty cycle is the period of time when all the power is supplied to the load.

A.2.2 PWM for spike-based processing

The way to use PWM to drive motors within a spiking neural network is to integrate the output spike signal until a fixed time TPWM and get the value to set for the PWM generator before the power stage. The PWM generator will set the duty cycle according to the integrated value.

A.3 PFM

A.3.1 Introduction

Pulse frequency modulation uses a squared waveform to carry the information, too. But in this case, the pulse width is fixed and the analog information is carried by the frequency. It means that the frequency of square wave will be modified according to the analog value to transmit. The amount of time when all the power is supplied to the load is fixed.

A.3.2 PFM for spike-based processing

The way to use PFM to drive motors within a spiking neural network is to spread the output spike signal to drive the motor until a fixed time λ and then, supply, straightforward, to the power stage. A couple of examples using PFM with spikes are in [41] and [8].

A.4 Comparative between both modulations

There are two big differences between these two modulations: the ripple on the response and the power consumption.

For the PWM, there is a trade-off between the time delay because of the integration TPWM time and the ripple this time introduce on the motor response (the motor is modeled as a low pass filter).

For PFM, the ripple introduced depends on the distribution of the spikes. Large 'silent times' should be avoided to minimize the ripple.

If we compare the power consumption, PFM modulation beats PWM. The reason is that for the PFM the duty cycle is fixed and for PWM it is modified according to the information and low in average.

A.5 Constraints

These two modulations are generated by a digital system such as a microcontroller or FPGA. Very often, these digital systems include PWM generators based on a register of n bits to codify the information. These 2n possibilities determine the amount of changes that can be done at the pulse width. For the PFM modulation, in principle, there is no such limitation. The maximum frequency just depends on the input signal frequency (defined by the clock of the digital system).

However, the power stage, typically based on natch bridges and opto-isolator components, limits the switching frequency up to 25 KHz or less. This frequency will be the maximum for the PFM modulation. Also the response time for the opto-isolator will fix the minimum pulse width for PFM.
If a frequency of 25 MHz is considered, to generate PWM, the maximum switching frequency means a period of 40 us for the maximum duty cycle (100% of the TPWM high).

A.6 Biological relationship

If we look through biological papers from neuroscientist, there is not a statement neither approaches to make a decision between PWM and PFM. But, there are some authors from engineer field that take as a fact that PFM is the most natural [122, 123].

Furthermore, if we consider biological data to make a decision:

The muscles are innervated by α and γ motor neurons (α motor neurons innervate extrafusal fibers and γ motor neurons innervate the intrafusal fibers) [4, 73]. The activity of the muscles has a relation with the frequency of the discharge to the motoneurons [3].

Thus, it looks like PFM is a good biological approach for motor controlling using spiking neural networks.

Appendix B

Publications

This appendix lists the publications that this Ph.D has produced.

B.1 Journal Papers

[1] Fernando Perez-Peña, Arturo Morgado-Estevez, Alejandro Linares-Barranco, Angel Jimenez-Fernandez, Fracisco Gomez-Rodriguez, Gabriel Jimenez-Moreno and Juan Lopez-Coronado. Neuro-inspired spike-based motion: from dynamic vision sensor to robot motor open-loop control through spike-VITE. Sensors, 13(11). pp. 15805-15832, 2013.

[2] Fernando Perez-Peña, Arturo Morgado-Estevez, Alejandro Linares-Barranco. Inter-spikes-intervals exponential and gamma distributions study of neuron firing rate for SVITE motor control model on FPGA. Neurocomputing, In Press, 2014.

B.2 Selected International Conferences

[1] Fernando Perez-Peña, Arturo Morgado-Estevez, Alejandro Linares-Barranco, Gabriel Jimenez-Moreno , Jose María Rodríguez-Corral and Rafael J. Montero-Gonzalez. Frequency Analysis of a 64x64 Pixel Retinomorphic System with AER Output to Estimate the Limits to Apply onto Specific Mechanical Environment. 11th International Work Conference on Artificial

Neural Networks, IWANN 2011, June 8-10, 2011.

[2] Fernando Perez-Peña, Arturo Morgado-Estevez, Rafael J. Montero-Gonzalez, Alejandro Linares-Barranco and Gabriel Jimenez-Moreno. Video Surveillance in industrial manufacturing machinery using an Address Event Vision sensor. Comparative between two different video sensor based on a bioinspired Retina. 8th International Joint Conference on e-Business and Telecommunications. ICETE 2011, July 18-21, 2011.

[3] Fernando Perez-Peña, Arturo Morgado-Estevez, Carlos Rioja-Del-Rio, Alejandro Linares-Barranco, Angel Jimenez-Fernandez, Juan Lopez-Coronado and Jose Luis Muñoz-Lozano. Towards AER VITE: building spike gate signal. 19th International Conference on Electronics, Circuits, and Systems (ICECS'2012), December 9-11, 2012.

[4] Fernando Perez-Peña, Arturo Morgado-Estevez, Alejandro Linares-Barranco, Angel Jimenez-Fernandez, Juan Lopez-Coronado and Jose Luis Muñoz-Lozano. A FPGA Spike-Based Robot Controlled with Neuro-inspired VITE. 12th International Work Conference on Artificial Neural Networks, IWANN 2013, June 12-14, 2013.

[5] Fernando Perez-Peña, Arturo Morgado-Estevez, Alejandro Linares-Barranco, Manuel Dominguez-Morales and Angel Jimenez-Fernandez. SVITE: a spike-based VITE Neuro-inspired robot controller. 20th International Conference on Neural Information Processing (ICONIP'2013), November 3-7, 2013.

[6] Fernando Perez-Peña, Arturo Morgado-Estevez, Teresa Serrano-Gotarredona, Francisco Gomez-Rodriguez, Victor Ferrer-Garcia, Angel Jimenez-Fernandez, Alejandro Linares-Barranco. Spike-based VITE control with dynamic vision sensor applied to an arm robot. 2014 IEEE International Symposium on Circuits and Systems (ISCAS), June, 2014.

[7] Fernando Perez-Peña, Alejandro Linares-Barranco, Elisabetta Chicca. An Approach to Motor Control for Spike-based Neuromorphic Robotics. IEEE Biomedical Circuits and Systems Conference (BIOCAS), October 2014.

References

[1] Hélene Paugam-Moisy and Sander Bohte. Computing with spiking neuron networks. In Grzegorz Rozenberg, Thomas Bäck, and JoostN. Kok, editors, *Handbook of natural computing*, pages 335–376. Springer Berlin Heidelberg, 2012.

[2] F. Gomez-Rodriguez, L. Miro-Amarante, F. Diaz-del Rio, A Linares-Barranco, and G. Jimenez. Real time multiple objects tracking based on a bio-inspired processing cascade architecture. In *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pages 1399–1402, May 2010.

[3] W. F. Ganong and K. E. Barrett. *Ganong's Review of Medical Physiology*. LANGE Basic Science. McGraw-Hill Medical, 24 edition, 2005.

[4] Arthur C. Guyton and John E. Hall. *TextBook of medical physiology*. Elsevier Saunders, 11 edition, 1961.

[5] Mei-Hong Qiu, Michael C. Chen, Zhi-Li Huang, and Jun Lu. Neuronal activity (c-fos) delineating interactions of the cerebral cortex and basal ganglia. *Frontiers in Neuroanatomy*, 8(13), March 2014.

[6] D. Bullock and S. Grossberg. *Volitional Action*, chapter VITE and FLETE: Neural modules for trajectory formation and postural control, pages 253–297. Elsevier, November 1989.

[7] Sonja Grün and Stefan Rotter. *Analysis of parallel spike trains*, volume 7. Springer, 2010.

[8] Fernando Perez-Peña, Arturo Morgado-Estevez, Alejandro Linares-Barranco, Angel Jimenez-Fernandez, Francisco Gomez-Rodriguez,

Gabriel Jimenez-Moreno, and Juan Lopez-Coronado. Neuro-inspired spike-based motion: from dynamic vision sensor to robot motor open-loop control through spike-vite. *Sensors*, 13(11):15805–15832, 2013.

[9] Giacomo Indiveri, Bernabé Linares-Barranco, Tara Julia Hamilton, André van Schaik, Ralph Etienne-Cummings, Tobi Delbruck, Shih-Chii Liu, Piotr Dudek, Philipp Häfliger, Sylvie Renaud, Johannes Schemmel, Gert Cauwenberghs, John Arthur, Kai Hynna, Fopefolu Folowosele, Sylvain Saighi, Teresa Serrano-Gotarredona, Jayawan Wijekoon, Yingxue Wang, and Kwabena Boahen. Neuromorphic silicon neuron circuits. *Frontiers in neuroscience*, 5(73), 2011.

[10] G. Indiveri, S. C. Liu, T. Delbrück, and R. Douglas. *Encyclopedia of Neuroscience*, chapter Neuromorphic Systems, pages 521–528. Academic Press, 2009.

[11] Kwabena Boahen. Neuromorphic microchips. *Scientific American*, 292(5):56–63, 2005.

[12] R. Douglas, M. Mahowald, and C. Mead. *Annual review of neuroscience*, pages 255–281.

[13] Carver Mead and Mohammed Ismail. *Analog VLSI implementation of neural systems*. Springer Science & Business Media, 1989.

[14] Wolfgang Maass. Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, 10(9):1659–1671, 1997.

[15] Cornelia Fermuller, Ralph Etienne-Cummings, Shih-Chii Liu, and Timothy Horiuchi. Telluride neuromorphic cognition engineering workshop. http://ine-web.org/telluride-conference-2014/.

[16] CapoCaccia Cognitive Neuromorphic Engineering Workshop. https://capocaccia.ethz.ch/capo/wiki/2014.

[17] Massimo Antonio Sivilotti. *Wiring considerations in analog VLSI systems, with application to field-programmable networks*. PhD thesis, California Institute of Technology, July 1990.

[18] Kwabena A Boahen. Point-to-point connectivity between neuromorphic chips using address events. *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, 47(5):416–434, May 2000.

[19] Elisabetta Chicca, Fabio Stefanini, Chiara Bartolozzi, and Giacomo Indiveri. Neuromorphic electronic circuits for building autonomous cognitive systems. *Proceedings of the IEEE*, pages 1–22, 2014.

[20] S Maya, R Reynoso, C Torres, and M Arias-Estrada. Compact spiking neural network implementation in fpga. In ReinerW. Hartenstein and Herbert Grünbacher, editors, *Field-Programmable Logic and Applications: The Roadmap to Reconfigurable Computing*, volume 1896 of *Lecture Notes in Computer Science*, pages 270–276. Springer Berlin Heidelberg, 2000.

[21] Liam P. Maguire, T. Martin McGinnity, Brendan Glackin, Arfan Ghani, Ammar Belatreche, and Jim Harkin. Challenges for large-scale implementations of spiking neural networks on fpgas. *Neurocomputing*, 71(1-3):13–29, 2007.

[22] Patrick Rocke, Brian McGinley, John Maher, Fearghal Morgan, and Jim Harkin. Investigating the suitability of fpaas for evolved hardware spiking neural networks. In GregoryS. Hornby, Lukas Sekanina, and PaulineC. Haddow, editors, *Evolvable Systems: From Biology to Hardware*, volume 5216, pages 118–129. Springer Berlin Heidelberg, 2008.

[23] Eugenio Culurciello, Ralph Etienne-Cummings, and Kwabena A. Boahen. A biomorphic digital image sensor. *Solid-State Circuits, IEEE Journal of*, 38(2):281–294, February 2003.

[24] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. A 128x128 120 db 15 us latency asynchronous temporal contrast vision sensor. *Solid-State Circuits, IEEE Journal of*, 43(2):566–576, 2008.

[25] Kwabena A. Boahen and Andreas G. Andreou. A contrast sensitive silicon retina with reciprocal synapses. volume 4, pages 764–772. Morgan Kaufmann, 1991.

[26] Juan Antonio Leñero Bardallo, Teresa Serrano-Gotarredona, and Bernabé Linares-Barranco. A 3.6 s latency asynchronous frame-free event-driven dynamic-vision-sensor. *Solid-State Circuits, IEEE Journal of*, 46(6):1443–1455, June 2011.

[27] Teresa Serrano-Gotarredona and Bernabé Linares-Barranco. A 128x128 1.5asynchronous frame-free dynamic vision sensor using transimpedance preamplifiers. *Solid-State Circuits, IEEE Journal of*, 48(3):827–838, 2013.

[28] Vincent Chan, Shih-Chii Liu, and André van Schaik. Aer ear: A matched silicon cochlea pair with address event representation interface. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 54(1):48–59, January 2007.

[29] Shih-Chii Liu, A Van Schaik, B.A Minch, and T. Delbruck. Event-based 64-channel binaural silicon cochlea with q enhancement mechanisms. In *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pages 2027–2030, May 2010.

[30] Thomas Jacob Koickal, Alister Hamilton, Su Lim Tan, James A Covington, Julian W Gardner, and Tim C Pearce. Analog vlsi circuit implementation of an adaptive neuromorphic olfaction chip. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 54(1):60–73, 2007.

[31] Chiara Bartolozzi and Giacomo Indiveri. Synaptic dynamics in analog vlsi. *Neural Computation*, 19(10):2581–2603, 2007.

[32] Giacomo Indiveri, Elisabetta Chicca, and Rodney Douglas. A vlsi array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity. *Neural Networks, IEEE Transactions on*, 17(1):211–221, 2006.

[33] Ben Varkey Benjamin, Peiran Gao, Emmett McQuinn, Swadesh Choudhary, Anand R. Chandrasekaran, Jean Marie Bussat, Rodrigo Alvarez-Icaza, John V. Arthur, Paul A. Merolla, and Kwabena Boahen. Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations. *Proceedings of the IEEE*, 102(5):699–716, 2014.

[34] Henry Markram, Karlheinz Meier, Thomas Lippert, Sten Grillner, Richard Frackowiak, Stanislas Dehaene, Alois Knoll, Haim Sompolinsky, Kris Verstreken, Javier DeFelipe, Seth Grant, Jean Pierre Changeux, and Alois Sariam. Introducing the human brain project. *Procedia Computer Science*, 7(0):39–42, 2011. Proceedings of the 2nd

European Future Technologies Conference and Exhibition 2011 (FET 11).

[35] Steve B. Furber, Francesco Galluppi, Steve Temple, and Luis A. Plana. The spinnaker project. *Proceedings of the IEEE*, 102(5):652–665, May 2014.

[36] Johannes Schemmel, D Bruderle, A Grubl, Matthias Hock, Karlheinz Meier, and Sebastian Millner. A wafer-scale neuromorphic hardware system for large-scale neural modeling. In *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pages 1947–1950. IEEE, May 2010.

[37] Andrew S. Cassidy, Paul Merolla, John V. Arthur, Steve K. Esser, Bryan Jackson, Rodrigo Alvarez-Icaza, Pallab Datta, Jun Sawada, Theodore M. Wong, Vitaly Feldman, Arnon Amir, Daniel Ben Dayan Rubin, Filipp Akopyan, Emmett McQuinn, William P. Risk, and Dharmendra S. Modha. Cognitive computing building block: A versatile and efficient digital neuron model for neurosynaptic cores. In *in International Joint Conference on Neural Networks (IJCNN). IEEE*, 2013.

[38] Jörg Conradt, Matthew Cook, Raphael Berner, Patrick Lichtsteiner, Rodney J Douglas, and T Delbruck. A pencil balancing robot using a pair of aer dynamic vision sensors. In *Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on*, pages 781–784. IEEE, May 2009.

[39] Christian Denk, Francisco Llobet-Blandino, Francesco Galluppi, LuisA. Plana, Steve Furber, and Jörg Conradt. Real-time interface board for closed-loop robotic tasks on the spinnaker neural computing system. In Valeri Mladenov, Petia Koprinkova-Hristova, Günther Palm, AlessandroE.P. Villa, Bruno Appollini, and Nikola Kasabov, editors, *Artificial Neural Networks and Machine Learning – ICANN 2013*, volume 8131 of *Lecture Notes in Computer Science*, pages 467–474. Springer Berlin Heidelberg, 2013.

[40] F. Perez-Peña, A. Morgado-Estevez, T. Serrano-Gotarredona, F. Gomez-Rodriguez, V. Ferrer-Garcia, A. Jimenez-Fernandez, and A. Linares-Barranco. Spike-based vite control with dynamic vision sensor applied to an arm robot. In *Circuits and Systems (ISCAS), 2014 IEEE International Symposium on*, pages 463–466, June 2014.

[41] Angel Jimenez-Fernandez, Gabriel Jimenez-Moreno, Alejandro Linares-Barranco, Manuel J. Dominguez-Morales, Rafael Paz-Vicente, and Anton Civit-Balcells. A neuro-inspired spike-based pid motor controller for multi-motor robots with low cost fpgas. *Sensors*, 12(4):3831–3856, 2012.

[42] Francesco Galluppi, Christian Denk, Matthias C Meiner, Terrence C Stewart, Luis A Plana, Chris Eliasmith, Steve Furber, and Jörg Conradt. Event-based neural computing on an autonomous mobile platform. *Brain*, 12:21.

[43] Fok-Sam Neckar Alex Khatib Oussama Menon, Samir and Kwabena Boahen. Controlling articulated robots in task-space with spiking silicon neurons. In *Proceedings of the IEEE International Conference on Biomedical Robotics and Biomechatronics*, August 2014.

[44] Fernando Perez-Peña, Alejandro Linares-Barranco, and Elisabetta Chicca. An approach to motor control for spike-based neuromorphic robotics. In *IEEE Biomedical Circuits and Systems Conference, 2014. BIOCAS*, Lausanne, 2014. IEEE.

[45] Chris Eliasmith. Cognition with neurons: A large-scale, biologically realistic model of the wason task. In *Proceedings of the XXVII annual conference of the cognitive science society*, pages 624–629, 2005.

[46] Chris Eliasmith. *How to build a brain: A neural architecture for biological cognition*. Oxford University Press, 2013.

[47] Terrence C Stewart, Feng-Xuan Choo, and Chris Eliasmith. Spaun: A perception-cognition-action model using spiking neurons. In *Proceedings of the 34th Annual Conference of the Cognitive Science Society*, 2012.

[48] Chris Eliasmith, Terrence C Stewart, Xuan Choo, Trevor Bekolay, Travis DeWolf, Yichuan Tang, and Daniel Rasmussen. A large-scale model of the functioning brain. *Science*, 338(6111):1202–1205, November 2012.

[49] R. Jung, E. J. Brauer, and J. J. Abbas. Real-time interaction between a neuromorphic electronic circuit and the spinal cord. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 9(3):319–326, September 2001.

[50] R. Jacob Vogelstein, Francesco Tenore, Lisa Guevremont, Ralph Etienne-Cummings, and Vivian K. Mushahwar. A silicon central pattern generator controls locomotion in vivo. *Biomedical Circuits and Systems, IEEE Transactions on*, 2(3):212–222, September 2008.

[51] Moisés Piedade, José Gerald, Leonel Augusto Sousa, Gonçalo Tavares, and Pedro Tomás. Visual neuroprosthesis: a non invasive system for stimulating the cortex. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 52(12):2648–2662, December 2005.

[52] E. Fernandez, F. Pelayo, S. Romero, M. Bongard, C. Marin, A. Alfaro, and L. Merabet. Development of a cortical visual neuroprosthesis for the blind: the relevance of neuroplasticity. *Journal of neural engineering*, 2(4):R1, 2005.

[53] Linares-Barranco A., R. Paz-Vicente, G. Jimenez, J.L. Pedreno-Molina, J. Molina-Vilaplana, and J. Lopez-Coronado. Aer neuro-inspired interface to anthropomorphic robotic hand. In *Neural Networks, 2006. IJCNN '06. International Joint Conference on*, pages 1497–1504, 2006.

[54] Emin Orhan. The leaky integrate-and-fire neuron model, November 2012.

[55] Fabio Stefanini, Emre Neftci, Sadique Sheik, and Giacomo Indiveri. Pyncs: a microkernel for high-level definition and configuration of neuromorphic electronic systems. *Frontiers in Neuroscience*, 8(73):1–14, August 2014.

[56] Rodney Douglas, Misha Mahowald, and Adrian Whatley. Communications infrastructure for neuromorphic analog vlsi systems. 1998.

[57] Rafael Serrano-Gotarredona, Matthias Oster, Patrick Lichtsteiner, Alejandro Linares-Barranco, Rafael Paz-Vicente, Francisco Gómez-Rodríguez, Luis Camuñas Mesa, Raphael Berner, Manuel Rivas-Pérez, and Tobi Delbruck. Caviar: A 45k neuron, 5m synapse, 12g connects/s aer hardware sensory–processing–learning–actuating system for high-speed visual object recognition and tracking. *Neural Networks, IEEE Transactions on*, 20(9):1417–1438, September 2009.

[58] Carlos Zamarreño Ramos, Rafael Serrano-Gotarredona, Teresa Serrano-Gotarredona, and Bernabé Linares-Barranco. Lvds interface

for aer links with burst mode operation capability. In *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*, pages 644–647. IEEE, May 2008.

[59] C. Zamarreño Ramos, Teresa Serrano-Gotarredona, Bernabé Linares-Barranco, Raghavendra Kulkarni, and Jose Silva-Martinez. Voltage mode driver for low power transmission of high speed serial aer links. In *Circuits and Systems (ISCAS), 2011 IEEE International Symposium on*, pages 2433–2436. IEEE, 2011.

[60] Hans Kristian Otnes Berge and Philipp Hafliger. High-speed serial aer on fpga. In *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*, pages 857–860. IEEE, May 2007.

[61] Abbott Laurence F. Dayan, Peter. *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. Computational Neuroscience. Massachusetts Institute of Technology Press, 6 edition, 2001.

[62] Paul Cisek. Neural representations of motor plans, desired trajectories, and controlled objects. *Cognitive Processing*, 6(1):15–24, 2005.

[63] D. M. Wolpert. Ted talk: The real reason for brains, 2011.

[64] Eric Kandel, James H. Schwartz, and Thomas M. Jessell. *Principles of Neural Science*. McGraw-Hill Medical, 4 edition, 2000.

[65] Yale E. Cohen and Richard A. Andersen. A common reference frame for movement plans in the posterior parietal cortex. *Nature Reviews. Neuroscience*, 3(7):553–562, July 2002.

[66] P. N. Sabes. The planning and control of reaching movements. *Current Opinion in Neurobiology*, 10(6):740–746, December 2000.

[67] A. P. Georgopoulos. Neural integration of movement: role of motor cortex in reaching. *The FASEB Journal*, 2(13):2849–2857, October 1988.

[68] Nicholas G. Hatsopoulos and Aaron J. Suminski. Sensing with the motor cortex. *Neuron*, 72(3):477–487, November 2011.

[69] D. J. Crammond and J. F. Kalaska. Prior information in motor and premotor cortex: activity during the delay period and effect on premovement activity. *Journal of Neurophysiology*, 84(2):986–1005, August 2000.

[70] Travis DeWolf. Noch: A framework for biologically plausible models of neural motor control. Master's thesis, University of Waterloo, January 2010.

[71] D. M. Wolpert, R. C. Miall, and M. Kawato. Internal models in the cerebellum. *Trends in Cognitive Sciences*, 2(9):338–347, September 1998.

[72] D. Bullock and S. Grossberg. Neural dynamics of planned arm movements: emergent invariants and speed-accuracy properties during trajectory formation. *Psychological Review*, 95(1):49–90, January 1988.

[73] Mark L. Latash. *Fundamentals of Motor Control*. Elsevier, 1 edition, 2012.

[74] R. Shadmehr and Steven P. Wise. *The computational neurobiology of reaching and pointing: a foundation for motor learning*. Massachusetts Institute of Technology Press, 2005.

[75] Ann M. Graybiel. The basal ganglia. *Current Biology*, 10(14):R509–R511, jul 2000.

[76] Ann M. Graybiel. The basal ganglia: learning new tricks and loving it. *Current Opinion in Neurobiology*, 15(6):638–644, December 2005.

[77] K. Doya. What are the computations of the cerebellum, the basal ganglia and the cerebral cortex? *Neural Networks*, 12(7-8):961–974, October 1999.

[78] K. Doya. Complementary roles of basal ganglia and cerebellum in learning and motor control. *Current Opinion in Neurobiology*, 10(6):732–739, December 2000.

[79] H. Mushiake and P. L. Strick. Pallidal neuron activity during sequential arm movements. *Journal of Neurophysiology*, 74(6):2754–2758, December 1995.

[80] Henk J. Groenewegen. The basal ganglia and motor control. *Neural Plasticity*, 10(1-2):107–120, 2003.

[81] E. Henneman. The size-principle: a deterministic output emerges from a set of probabilistic connections. *The Journal of Experimental Biology*, 115:105–112, March 1985.

[82] R. A. Conwit, D. Stashuk, B. Tracy, McHugh M., Brown W.F., and Metter E.J. The relationship of motor unit size, firing rate and force. *Clinical Neurophysiology*, 110(7):1270–1275, July 1999.

[83] C. J. De Luca. Control properties of motor units. *The Journal of Experimental Biology*, 115:125–136, March 1985.

[84] M. C. Binder, P. Bawa, P. Ruenzel, and E. Henneman. Does orderly recruitment of motoneurons depend on the existence of different types of motor units? *Neuroscience Letters*, 36(1):55–58, March 1983.

[85] Youngnam Kang, Mitsuru Saito, Hiroki Toyoda, and Hajime Sato. *Progress in brain research*, volume 187, chapter Recruitment of masseter motoneurons by the presumed spindle Ia inputs, pages 163–171. Elsevier, January 2010.

[86] T. C. Cope and A. J. Sokoloff. Orderly recruitment among motoneurons supplying different muscles. *Journal of Physiology, Paris*, 93(1-2):81–85, 1999.

[87] Terrence C. Stewart, Bryan Tripp, and Chris Eliasmith. Python scripting in the nengo simulator. *Frontiers in Neuroinformatics*, 3(7), 2009.

[88] Dan F. Goodman and Romain Brette. The brian simulator. *Frontiers in Neuroscience*, 3(2):192–197, September 2009.

[89] Michael L. Hines and Nicholas T. Carnevale. The neuron simulation environment. *Neural Computation*, 9(6):1179–1209, August 1997.

[90] Simulation of networks of spiking neurons: a review of tools and strategies. *Journal of Computational Neuroscience*, 23(3):349–398, December 2007.

[91] P. Gaudiano and S. Grossberg. Vector associative maps: Unsupervised real-time error-based learning and control of movement trajectories. *Neural Networks*, 4(2):147–183, 1991.

[92] Daniel Bullock, Stephen Grossberg, and Frank H. Guenther. A self-organizing neural model of motor equivalent reaching and tool use by a multijoint arm. *Journal of Cognitive Neuroscience*, 5(4):408–435, October 1993.

[93] J. S. Albus. A new approach to manipulator control: The cerebellar model articulation controller (cmac). *Journal of Dynamic Systems, Measurement, and Control*, 97(3):220–227, September 1975.

[94] Micha Hersch and Aude G. Billard. *Autonomous Robots*, volume 25, chapter Reaching with multi-referential dynamical systems, pages 71–83. Springer US, January 2008.

[95] Emanuel Todorov and Michael I. Jordan. Optimal feedback control as a theory of motor coordination. *Nature Neuroscience*, 5(11):1226–1235, November 2002.

[96] Emanuel Todorov. Optimality principles in sensorimotor control. *Nature Neuroscience*, 7(9):907–915, September 2004.

[97] U. Pattacini, F. Nori, L. Natale, G. Metta, and G. Sandini. An experimental evaluation of a novel minimum-jerk cartesian controller for humanoid robots. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1668–1674. IEEE, October 2010.

[98] Micha Hersch and Aude G. Billard. A biologically-inspired controller for reaching movements. In *Biomedical Robotics and Biomechatronics, 2006. BioRob 2006. The First IEEE/RAS-EMBS International Conference on*, pages 1067–1072. IEEE, February 2006.

[99] H. Nagasaki. Asymmetric velocity and acceleration profiles of human arm movements. *Experimental Brain Research*, 74(2):319–326, 1989.

[100] Daniel Bullock, Jose L. Contreras-Vidal, and Stephen Grossberg. Equilibria and dynamics of a neural network model for opponent muscle control. In GeorgeA. Bekey and KennethY. Goldberg, editors, *Neural Networks in Robotics*, volume 202 of *The Springer International Series in Engineering and Computer Science*, pages 439–457. Springer US, 1993.

[101] Katsuhiko Ogata and Yanjuan Yang. *Modern control engineering.* Prentice Hall, 5 edition, 1970.

[102] F. Perez-Peña, A Morgado-Estevez, C. Rioja-Del-Rio, A Linares-Barranco, A Jimenez-Fernandez, J. Lopez-Coronado, and J.L. Munoz-Lozano. Towards aer vite: Building spike gate signal. In *Electronics, Circuits and Systems (ICECS), 2012 19th IEEE International Conference on*, pages 881–884. IEEE, Dec 2012.

[103] Angel Jimenez-Fernandez. *Diseño y evaluación de sistemas de control y procesamiento de señales basados en modelos neuronales pulsantes.* PhD thesis, University of Seville, 2010.

[104] Z. F. Mainen and T. J. Sejnowski. Reliability of spike timing in neocortical neurons. *Science*, 268(5216):1503–1506, June 1995.

[105] W. R. Softky and C. Koch. The highly irregular firing of cortical cells is inconsistent with temporal integration of random epsps. *The Journal of neuroscience*, 13(1):334–350, 1993.

[106] T. Iakymchuk, A Rosado, T. Serrano-Gotarredona, B. Linares-Barranco, A Jimenez-Fernandez, A Linares-Barranco, and G. Jimenez-Moreno. An aer handshake-less modular infrastructure pcb with x8 2.5gbps lvds serial links. In *Circuits and Systems (ISCAS), 2014 IEEE International Symposium on*, pages 1556–1559, June 2014.

[107] Elena Cerezuela-Escudero, ManuelJesus Dominguez-Morales, Angel Jiménez-Fernández, Rafael Paz-Vicente, Alejandro Linares-Barranco, and Gabriel Jiménez-Moreno. Spikes monitors for fpgas, an experimental comparative study. In Ignacio Rojas, Gonzalo Joya, and Joan Gabestany, editors, *Advances in Computational Intelligence*, volume 7902 of *Lecture Notes in Computer Science*, pages 179–188. Springer Berlin Heidelberg, 2013.

[108] R. Berner, T. Delbruck, A Civit-Balcells, and A Linares-Barranco. A 5 meps $100 usb2.0 address-event monitor-sequencer interface. In *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*, pages 2451–2454, May 2007.

[109] Tobi Delbruck. jaer open source project. http://jaer.wiki.sourceforge.net/.

[110] Alejandro Linares-Barranco, Gabriel Jimenez-Moreno, Bernabé Linares-Barranco, and Antón Civit-Balcells. On algorithmic rate-coded aer generation. *IEEE Transactions on Neural Networks*, 17(3):771–788, May 2006.

[111] Giorgio Metta, Lorenzo Natale, Francesco Nori, and Giulio Sandini. Force control and reaching movements on the icub humanoid robot. In *Proceedings of 15th International Symposium on Robotics Research*, pages 1–21, 2011.

[112] Mathworks documentation center. http//www.mathworks.es.

[113] D.B. Fasnacht and G. Indiveri. A pci based high-fanout aer mapper with 2 gib ram look-up table, 0.8 us latency and 66mhz output event-rate. In *Information Sciences and Systems (CISS), 2011 45th Annual Conference on*, pages 1–6, March 2011.

[114] D.B. Fasnacht, AM. Whatley, and G. Indiveri. A serial communication infrastructure for multi-chip address event systems. In *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*, pages 648–651, May 2008.

[115] D. Beamish, I. Scott MacKenzie, and Jianhong Wu. Speed-accuracy trade-off in planned arm movements with delayed feedback. *Neural Networks*, 19(5):582–599, 2006.

[116] R. W. Rochelle. Pulse-frequency modulation. *Space Electronics and Telemetry, IRE Transactions on*, (2):107–111, 1962.

[117] Leon W. Couch, Muralidhar Kulkarni, and U. Sripati Acharya. *Digital and analog communication systems*, volume 6. Prentice Hall, 1997.

[118] D. Grahame Holmes and Thomas A. Lipo. *Pulse width modulation for power converters: principles and practice*, volume 18. John Wiley & Sons, 2003.

[119] Pragasen Pillay and Ramu Krishnan. Modeling, simulation, and analysis of permanent-magnet motor drives. ii. the brushless dc motor drive. *Industry Applications, IEEE Transactions on*, 25(2):274–279, 1989.

[120] J. D. Irwin, Marian P. Kazmierkowski, Ramu Krishnan, and Frede Blaabjerg. *Control in power electronics: selected problems*. Academic press, 2002.

[121] Google patent search. www.google.com/patents.

[122] Elliott J. Bayly. Spectral analysis of pulse frequency modulation in the nervous systems. *Biomedical Engineering, IEEE Transactions on*, (4):257–265, 1968.

[123] R. W. Jones, C. C. Li, A. U. Meyer, and R. B. Pinter. Pulse modulation in physiological systems, phenomenological aspects. *Bio-Medical Electronics, IRE Transactions on*, 8(1):59–67, 1961.